Spécif n° 21

Novembre 1992

Société des Personnels Enseignants et Chercheurs en Informatique de France, ENS, 45 rue d'Ulm - 75005 PARIS

SOMMAIRE

Société des Personnels Enseignants et Chercheurs en Informatique de France

SOMMAIRE

Société des Personnels Enseignants et Chercheurs en Informatique de France

CONSEIL D'ADMINISTRATION DE SPECIF
(1992)

Anciens Présidents	:	PAIR C. (1986-1988) COMYN G. (1989) CARREZ CH. (1990-1991)
Président	:	GIRAULT C.
Vice-Président	:	ARNOLD A.
Membres du C.A.	:	BARTHET M.F. BOYAT J., (Bureau), Rapporteur Commission Enseignement CHABRE-PECCOUD M., (Bureau), Rapporteur Commission Matériel-Logiciel COT N., (Bureau), Responsable des bulletins et des archives FLECK J. HERVIER Y., (Bureau), Trésorier HORLAIT ERIC, Président Commission Personnel JOURDAN M., Correspondant INRIA JULLIAND J., Membre Commission Enseignement LAFON P. , Membre Commission Enseignement LESCANNE P., Président Commission Recherche LUCAS M., Membre Commission Enseignement MARCENAC P. MOSSIERE J., Membre Commission Recherche RICHIER J.-L. DE SABLET G., Président Commission Matériel-Logiciel SCHNEIDER M., Diffusion du Bulletin SIROUX J.-P. STEEN J.-P., (Bureau), Secrétaire TOURNIER E., (Bureau), Rapporteur Commission Recherche VIGNOLLE J.
Bulletin Spécif	:	Editeur : COT N.
ADRESSE	:	Bulletin SPECIF N. COT EHEI 45, rue des Saints-Pères 75006 PARIS

(Le bulletin est imprimé et diffusé par M. SCHNEIDER)

VIE DE L'ASSOCIATION

. Compte rendu du Conseil d'Administration du 11 juin 1992

. Convocation de l'Assemblée Générale de Spécif

. 2èmes Journées de travail (Rennes, 25-26 mars 1992)

. Commission Recherche

SPECIF

CONSEIL D'ADMINISTRATION
du 11 Juin 1992 à Paris

Ont participé,

Membres: Cl. GIRAULT, J.P. STEEN, M. HERVIER, Ch. CARREZ, M. CHABRE-PECCOUD, N. COT, E. TOURNIER, E. HORLAIT, P. LESCANNE, J. MOSSIERE, G. de SABLET, M. SCHNEIDER, J. VIGNOLLE.

Invités : J.P. FINANCE (DRED), M. DAUCHET (CNRS), M. BIDOIT (MRE).

Excusés : J. BOYAT, M. JOURDAN, J. FLECK, J. SIROUX, M. LUCAS, M.F. BARTHET, M. JOURDAN, J. JULLIAND, P. LAFON, P. MARCENAC, J.L. RICHIER.

1. Décès de Mme de SABLET.

L'épouse de notre collègue est décédée dans un accident de circulation. En début de séance, le Conseil a observé une minute de silence. G. de SABLET remercie SPECIF et les collègues qui lui ont manifesté tant de sympathie pendant ces moments douloureux.

2. Exposés de nos collègues du Ministère.

L'objet principal de ce conseil était de rencontrer nos collègues qui ont des responsabilités dans les ministères. Chacun a présenté le domaine qu'il a en charge, puis une discussion a eu lieu. Plus que l'exposé, dont seuls quelques chiffres ont été sortis, c'est la discussion qui est reproduite ici.

1°) J.P. FINANCE (DRED).

Il est conseiller pour l'Informatique dans le SPI (Sciences Pour l'Ingénieur) qui est la DS4 (Direction Scientifique) de la DRED (Direction de

Finances. La décision est difficile à obtenir, surtout en période de restriction de budget. D'autre part, la répartition dans les laboratoires et les universités est très variable.

La proposition de création de poste d'ITA occupés par des jeunes en thèse, formule comparable aux ATER, reçoit un avis mitigé, en raison de ce que les ITA sont la mémoire de l'équipe et qu'ils doivent rester en poste pour de longues périodes.

Les **administratifs** sont en nombre insuffisant et trop occupés par les enquêtes et la "jonglerie" budgétaire. Ils seraient bien mieux utilisés si on allégeait ces charges. On se pose la question de l'utilité des enquêtes, plusieurs d'entre elles ont demandé la même information. La jonglerie budgétaire résulte de la dispersion des origines des financements et des règles de la comptabilité publique. Une seule source de financement simplifierait ce problème pratique, mais le CA considère qu'il y a plus d'avantage à avoir plusieurs sources de financement. Septicisme quant à l'efficacité d'un logiciel de gestion financière des laboratoires.

Parmi les actions spécifiques de la DRED, signalons l'**Ecole des Jeunes Chercheurs** (vers avril, depuis 2 ans maintenant). Elle permet beaucoup d'échanges entre eux. Le CA pense que le Bulletin devrait en publier le bilan.

La DRED ne peut **financer une association**, même SPECIF, mais il est envisageable de présenter des factures pour des réalisations particulières au profit du MEMC, comme par exemple une étude ou une enquête.

Le CA décide qu'une *synthèse des chiffres des tableaux de M. FINANCE, en vue de sortir des indicateurs* relatifs à notre discipline sera établie par la Commission Recherche en vue d'être publiée dans le Bulletin exceptionnel.

2°) M. DAUCHET (CNRS).

Il est Chargé de Mission pour l'Informatique dans la section 07 du secteur SPI du CNRS. Ce secteur représente 9,8 % du CNRS. L'Informatique rassemble à 30 à 35 unités et plus de 150 chercheurs.

En moyenne un laboratoire contient 5 Chercheurs CNRS, 30 enseignant-chercheurs et reçoit entre 2,5 et 3 MF de contrats. Les proportions CNRS/MEN des personnels sont, approximativement, 1/12 en Informatique, 1/6 au SPI, 1/3 au CNRS.

L'INRIA est équivalent en poids, à l'Informatique du CNRS et ne peut donc être ignoré. Sa prise en compte ramène ces rapports à des pourcentages plus raisonnables.

Un chercheur reçoit en moyenne 20 KF/an, brut. Mais le CNRS a une très étroite marge de manœuvre pour mener une politique globale au CNRS. Ainsi le SPI dispose d'environ 10 MF pour les actions ciblées. La direction du SPI et le CNRS, en général, tentent de dégager des moyens à ce niveau.

Pour les **ITA**, il y a eu beaucoup de nominations dans les années 50 au CNRS. Ces personnes vont partir à la retraite. Ceci devrait permettre une nouvelle répartition. Il faut être vigilant. Les autres disciplines manquent aussi d'ingénieurs, et nous sollicitent souvent pour être des "substituts" d'Ingénieurs. Il est réaffirmé que la "Recherche en Informatique est celle qui apporte quelque chose à la discipline".

Les autres disciplines méconnaissent la recherche en informatique confondant, par exemple, la programmation et le génie logiciel. Ceci résulte d'un manque de communication et de publicité autour de nos résultats et aussi de la conception même de l'Informatique qu'ont acquis nos collègues des autres disciplines dans leur séjour comme étudiant à l'Université. Il est donc bon de soigner l'enseignement de l'Informatique en DEUG.

La communauté Informatique devrait réfléchir à de "grands" programmes renforçant sa visibilité et son identité.

3°) M. BIDOIT (MRE).

Il est conseiller au MRE (Ministère de la Recherche et de l'Espace) dans le Département Mathématiques et Technologie de l'Information qui contient l'Informatique (mais pas la robotique).

La recherche n'est plus priorité nationale et le budget de département, de 75 MF, est passé à 62 MF de 91 à 92, tandis que l'Informatique passait de 36,5 à 30 MF. Il y a donc un gradient négatif, mais la réforme des **PRC** ne peut se faire que dans l'optique d'un budget globalement en diminution.

Par contre, l'enquête de SPECIF met en évidence certaines critiques. Les équipes de direction sont juges et parties, ceci ne permet pas une réelle politique sélective. Il y a nécessité d'un aménagement sur les structures de gestion scientifique.

Les frontières des PRC ne sont pas nettes. Elles correspondes à une certaine logique scientifique. Il est suggéré de développer des actions transversales.

Dans le prochain budget (celui de 92 qui va être publié bientôt), il y aura certainement différentiation entre financement de base et d'animation et celui des actions spécifiques. Le MRE va donc aussi travailler par actions incitatrices (opérations finalisées visibles). Le souhait de la communauté est d'avoir son mot à dire dans la définition de ces actions.

Les PRC favorisent la **pérennité de la recherche** à l'opposé des actions incitatrices. les nouveaux thèmes ont mis en évidence la vitalité de la communauté. Il y a cependant un ménage intérieur à faire.

Les appels d'offres ont pris beaucoup de retard. Quand ils paraîtront, les délais de réponse seront très courts. *Il est suggéré d'utiliser la messagerie pour les pré-diffuser et même d'utiliser le serveur SPECIF.* Y. HERVIER étudie cette affaire.

3. Questions diverses.

1. CNU

La liste de qualification des Professeurs est parue au Minitel, retard pour celle des Maîtres de Conférences en raison de la masse importante de dossiers à traiter.

2. Lettre de Mr TOMBRE.

Elle est en rapport avec la constitution des listes pour les élections au CNU.

Une réponse précise peut lui être envoyée, compte tenu de la position connue, et publiée dans le Bulletin de SPECIF.

Cette lettre a suscité une discussion sur le fait que les plus jeunes ne s'impliquent pas dans la gestion de SPECIF.

4. Informations du Président.

1. Mr Martin JOURDAN (INRIA) nous fait part de son intention de démissionner du CA à la prochaine AG. Le Conseil en prend acte et souhaite qu'une autre personne de l'INRIA soit candidate pour être le représentant de l'INRIA au CA.

2. Mr GIRAULT a été interviewé par un reporter de l'Express.

3. Il faut revoir l'adresse de SPECIF (actuellement à l'EN, rue d'Ulm). Il semble qu'il y ait des lettres qui se perdent.

5. Compte-rendu des CA précédents.

Ils sont approuvés, y compris celui du CA du 14 Mai 92.

6. Prochaines réunions.

C.A. Jeudi 15 Octobre 92 13h30 EHEI (Préparation A.G)

A.G.O. Jeudi 10 Décembre 92

specif

Société des Personnels Enseignants et Chercheurs en Informatique de France

Convocation de l'Assemblée Générale Ordinaire

Vous êtes convié(e) à la réunion de l'Assemblée Générale Ordinaire de SPECIF qui aura lieu le

Vendredi 11 décembre 1992
de 14h à 17h

Le lieu vous sera communiqué ultérieurement dans un courrier qui contiendra aussi bulletins de vote et pouvoirs.

Cette réunion se tiendra après les Journées de l'INRIA qui se dérouleront, les 10 et 11 décembre 1992, dans les locaux du MRT, rue Descartes à Paris. Ces journées se terminent à 11h45.

L'ordre du jour de l'Assemblée Générale sera :

Rapport Moral.
Rapport Financier.
Rapports des Commissions.
Vote du Quitus au Conseil d'Administration et au Président.
Elections au Conseil d'Administration.
Toute question adressée par écrit au Président 20 jours avant la date de l'Assemblée, et accompagnée des signatures de 40 Membres de SPECIF.

Si vous ne pouvez participer à l'Assemblée Générale, vous pourrez vous faire représenter par un Membre de SPECIF (à qui vous devez remettre un pouvoir) et/ou voter par correspondance (uniquement pour les élections au Conseil d'Administration).

Pour participer à l'Assemblée Générale, il faut être Membre actif de SPECIF, à savoir, enseignant ou chercheur en Informatique de l'Enseignement Supérieur ou d'un organisme de recherche publics, à jour de sa cotisation (120 Frs). On peut payer la cotisation à l'entrée de la salle de réunion ou l'envoyer à Y. HERVIER, Universite de Nice Sophia-Antipolis, Parc Valrose, 06034 NICE Cedex.

SPECIF

Conseil d'Administration.

(AGO du 5.12.91)

Les membres **sortants** en 1992 sont marqués d'une *.

Bureau		
GIRAULT Claude (Président)	Paris 6	
ARNOLD André (Vice Président)	Bordeaux 1	*
STEEN Jean-Pierre (Secrétaire)	Lille 1	*
HERVIER Yves (Trésorier)	Nice	
BOYAT Janine	IUT - Montpellier	*
CARREZ Christian	CNAM -Paris	*
CHABRE-PECCOUD Monique	Grenoble	
COT Norbert	EHEI -Paris	*
TOURNIER Evelyne	Grenoble	
Assesseurs :		
BARTHET Marie France	Toulouse	
FLECK Jacques	IUT - Strasbourg	
HORLAIT Eric	Paris 6	
JOURDAN Martin	INRIA - Rocquencourt	
JULLIAND Jacques	Besançon	
LAFON Pierre	IUT - Bordeaux	
LESCANNE Pierre	CNRS - Nancy	
LUCAS Michel	ECN - Nantes	*
MARCENAC Pierre	La Réunion	
MOSSIERE Jacques	ENSIMAG - Grenoble	
RICHIER Jean-Luc	CNRS - Grenoble	*
de SABLET Georges	IUT - Paris	
SCHNEIDER Michel	Clermont 2	
SIROUX Jacques	IUT - Lannion	
VIGNOLLE Jean	Toulouse	*

SPECIF

Conseil d'Administration.

APPEL de CANDIDATURES

(AGO du 11.12.92)

Date limite de dépôt des candidatures : **13 Novembre 1992**

Le Conseil d'Administration de SPECIF est composé de 24 administrateurs, renouvelables par tiers tous les ans. Cette année encore, 8 membres du conseil dont le mandat de 3 ans arrive à terme, seront sortants et devront être renouvelés ou remplacés par élection lors de l'Assemblée Générale du 11 décembre 1992. Pouvant assurer jusqu'à trois mandats, tous sont rééligibles.

Le vote par correspondance pour l'élection des nouveaux administrateurs sera possible. De ce fait, la liste des candidats devra être connue avant la réunion. Il est donc nécessaire de faire acte de candidature. C'est ce que nous vous demandons ici.

Tout membre actif* peut être élu au Conseil d'Administration. Pour faire acte de candidature, il suffit d'adresser au secrétaire le document ci-joint après l'avoir rempli.

Merci à ceux qui, mesurant l'intérêt de l'association pour la Communauté Informatique, acceptent de participer à son administration.

Le Secrétaire.

* Note : Est membre actif de SPECIF, tout enseignant ou chercheur en Informatique de l'enseignement supérieur ou d'un organisme de recherche publics, à jour de sa cotisation (120 Frs). On peut joindre la cotisation à l'acte de candidature.

Candidature à adresser au Secrétaire de SPECIF, au plus tard, le 13 Novembre 1992:

J.P. STEEN LIFL- UFR d'IEEA - USTL (Lille 1) - 59655 VILLENEUVE D'ASCQ CEDEX
Tel professionnel : 20 43 42 60 - Tel secrétariat: 20 43 47 24 - Télécopie : 20 43 65 66
Messagerie : steen@FRCITL81.bitnet

SPECIF

Candidature au Conseil d'Administration.

(AGO du 11.12.92)

Date limite de dépôt des candidatures : **13 Novembre 1992**

NOM :.....
Prénom :.....
Profession :.....
Adresse professionnelle :.....
.....
.....

déclare être candidat au Conseil d'Administration de SPECIF.

A ,le

Signature :

Curriculum vitae succinct ° : .....
.....
.....
.....

Profession de foi (facultatif)° :.....
.....
.....
.....

Pour vous joindre :

Téléphone personnel* :.....
Téléphone professionnel* :.....
Téléphone secrétariat* :.....
Télécopie (Fax)*.....
Messagerie (email)* :.....

Renseignements complémentaires (pour la déclaration à la Préfecture, en cas d'élection):

Date et lieu de naissance* :.....
Nationalité* :.....
Domicile* :.....
.....
.....

° Sera publié avec les bulletins de vote.

* Indiquer si cette information ne doit pas être publiée, voire même communiquée aux autres Membres du Conseil d'Administration.

Candidature à adresser au Secrétaire de SPECIF, au plus tard, **le 13 Novembre 1992:**

J.P. STEEN LIFL- UFR d'IEEA - USTL (Lille 1) - 59655 VILLENEU' 'E D'ASCQ CEDEX

Tel professionnel : 20 43 42 60 - Tel secrétariat: 20 43 47 24 - Télécopie : 20 43 65 66

Messagerie : steen@FRCITL81.bitnet

2èmes Journées de travail

Les langages applicatifs dans l'enseignement de l'informatique

Rennes (IFSIC / IRISA) les 25-26 Mars 1993

Co-organisées par Spécif

Comité de programme

Michel Briand (ENST Bretagne)
Guy Cousineau (ENS, Ulm)
Daniel Herman (IFSIC, IRISA)
Christian Queinnec (Polytechnique-INRIA)
Jean-Claude Royer (Université de Nantes)
Harald Wertz (Université de St Denis)
Pierre Casteran (Université de Bordeaux)
Véronique Donzeau-Gouge (CNAM-INRIA)
Jean-François Perrot (Laforia)
Daniel Ribbens (Université de Liège)
Emmanuel Saint James (Bull, LITP)

Organisation

Michel Briand (inscription)
(tél : 98 00 12 80, fax 98 00 12 82)
briand@enstb.enst-bretagne.fr

Daniel Herman, Sophie Robin (Rennes)
(tél 99 84 71 76, fax 99 38 38 32)
herman@irisa.fr, robin@irisa.fr

texte de présentation des journées au verso

Bulletin d'inscription

Nom : ..... Adresse : .....

Prénom : ..... Tél : .....

Organisme : ..... E-mail : .....

s'inscrit aux journées de travail sur les langages applicatifs dans l'enseignement de l'informatique.

A ..... le ..... signature.....

Droits d'inscription 350 Francs (incluant les déjeuners et les actes) à joindre à la fiche (chèque ou bon de commande à l'ordre de *promotion et essor scientifique*)

Un courrier concernant le programme et les modalités pratiques d'accueil sera envoyé début 93 à toutes les personnes inscrites.

Fiche d'inscription à retourner à :

Télécom Bretagne
Secrétariat dépt Informatique et Réseaux
BP 832 29 285 Brest Cédex

tel : 98 00 12 80
fax : 98 00 12 82

Les langages applicatifs dans l'enseignement de l'informatique

L'enseignement de l'informatique en tant que discipline dans les formations de base conduit un nombre croissant d'universités et d'écoles à expérimenter l'utilisation de langages fonctionnels.

Les langages fonctionnels tels que Lisp, Scheme, ML, Miranda constituent en effet un terreau fertile propre à former la base d'une culture informatique. L'introduction des fonctions, objets à part entière, permet d'appréhender des phénomènes profonds de l'informatique sur des exem-ples de taille raisonnable. La notion de fonction constitue un repère fixe dans un environnement matériel perpétuellement changeant. Plus proches des mathématiques et syntaxiquement non encombrés, les langages applicatifs laissent une large place à l'imagination ainsi qu'à l'expérimentation de nouveaux procédés et/ou codages. Leurs dérivés parmi lesquels il faut compter les systèmes de calcul formel permettent même un renouvellement de la pédagogie des mathématiques. Après le succés d'une première journée (MRT Paris, 20 mars 91) et la multiplication des expériences (rencontre de Rennes novembre 91), nous organisons les deuxièmes journées sur les langages applicatifs dans l'enseignement de l'informatique les 25 et 26 mars 93 à Rennes.

Cette rencontre a pour but, d'une part, de présenter les diverses voies explorées, de favoriser la diffusion des expériences et des matériels pédagogiques réalisés. D'autre part ces journées permettront de débattre sur l'endroit exact où les différents concepts de l'informatique doivent être introduits dans le cursus scolaire, de leur articulation et du lien avec l'enseignement des autres disciplines.

Nous souhaitons recevoir des communications présentant des recherches sur les apports des langages applicatifs dans l'enseignement de l'informatique ou relatant et analysant des expériences selon cettte approche,

Nous sommes particulièrement intéressés par l'utilisation des langages applicatifs dans le cadre de la formation initiale à l'informatique.

2èmes Journées de travail

Les langages applicatifs dans l'enseignement de l'informatique

Rennes (IFSIC / IRISA)

les 25-26 Mars 1993

Commission Recherche de SPECIF

COMPTE RENDU DE LA RÉUNION TÉLÉPHONIQUE DU 23 SEPTEMBRE 1992

PARTICIPANTS : André ARNOLD, Jean-Claude BERMOND, Pierre LESCANNE, Brigitte ROZOY, Dominique SOTTEAU, Evelyne TOURNIER.

RELEVÉ DE LA PRÉSIDENCE DE LA COMMISSION RECHERCHE -

Devant prendre des responsabilités au plan national, Pierre LESCANNE souhaite se décharger de la fonction de Président de la Commission recherche. Brigitte ROZOY ayant accepté de prendre la relève, il est proposé une succession en biseau avec évidemment passage par les instances statutaires de SPECIF. Pierre LESCANNE continuera à faire partie de la Commission recherche de SPECIF. Brigitte ROZOY étant actuellement secrétaire de l'Association Française d'Informatique Théorique (AFIT), un débat s'engage sur le problème de ce cumul de mandat. Au lieu d'être un problème, cette double fonction peut ouvrir à une coopération avec les associations de chercheurs de l'informatique : AFIA, AFIT, SPECIF pour une représentativité accrue auprès des pouvoirs publics à défaut de l'existence d'une société savante pluraliste effective. La Commission recherche SPECIF restera cependant l'endroit où seront évoqués les problèmes relevant spécifiquement de l'exercice de la profession de chercheur en informatique.

De plus, il faut que le Président de SPECIF et le Président de la Commission recherche rencontrent les directeurs des établissements de tutelle de la recherche en informatique (BAIXERAS au MRT, DESCUSSE à la DRED, GAGNEPAIN au CNRS). C'est une idée qui avait été évoquée lors de précédentes réunions et qui n'a pu être mise en oeuvre. Une décision du Conseil d'administration en ce sens sera demandée.

COMITÉ THÉMATIQUE -

Une discussion a eu lieu entre André ARNOLD, Jean-Claude BERMOND et Pierre LESCANNE pour préparer la réunion du lendemain au Quai Anatole France (siège du CNRS) que Jean-Claude BERMOND préside et à laquelle Pierre LESCANNE participera. Cette réunion a pour but de donner une prospective au département SPI du CNRS dans le domaine des sciences de l'information.

Pierre LESCANNE

NOUVELLES DU C.N.U.

. Composition de la 61ème section

. Session de JUIN 1992 de la 61ème section

. Promotions 92 (27 section)

Composition de la 61ème section

SECTION 61 : GÉNIE INFORMATIQUE, AUTOMATIQUE ET TRAITEMENT DU SIGNAL

COLLEGE 1

Noms, Prénoms
ABOU-KANDIL Hisham
ARQUES Pierre-Yves
AUBRUN Michel
BERTRAND Jean-Claude
BINDER Gérard
BORNE Pierre
CORRAZZA Michel
DEMOMENT Guy
DERNIAME Jean-Claude
DUBUISSON Bernard
FOURNIER Alain
JUANOLE Guy
JUTARD Alain
KHALIL Wisama
LADET Pierre
LATTUATI Vincent
OUSTALOUP Alain
SOENEN René

COLLEGE 2

Noms, Prénoms
AULOGE Jean-Yves
BAUDOIS Daniel
BERGEON Benoît
BOLON Philippe
BONTON Pierre
COEURDEUIL Gilles
DESCOTES GENON Bernard
DURAND Serge
ELLOY Jean-Pierre
ESTEBAN Philippe
FANTON Jean-Claude
GARETTE GRAU Brigitte
MAGNIN Isabelle
MONGENET Catherine
REYNAUD Roger
RICHARD Alain
ROSSETTO Bruno
RUBEL Paul

61ème section du CNU : Génie informatique, automatique et traitement du signal

COMPTE RENDU DE LA SESSION DE JUIN 1992 INSCRIPTION SUR LA LISTE DE QUALIFICATION AUX FONCTIONS DE PROFESSEUR

RÉSULTATS -

150 candidats.
81 inscrits sur la liste de qualification.
68 non inscrits dont 3 pour des raisons administratives (dossier non parvenu par exemple).

ÉLÉMENTS QUALITATIFS -

Le travail de la Commission était d'évaluer, à partir des dossiers fournis par les candidats, leurs aptitudes à remplir les fonctions de Professeur. Un candidat standard est censé :

- avoir une expérience d'enseignement supérieur significative,
- avoir effectué une recherche de qualité, attestée par des publications de bon niveau dans des revues et des congrès avec comité de lecture,
- avoir encadré des jeunes chercheurs,
- avoir créé un axe de recherche et/ou animé une petite équipe de recherche.

Il peut également, mais sans que cela soit nécessaire, avoir déjà assumé des responsabilités collectives.

Un Maître de conférences ayant obtenu l'Habilitation à diriger les recherches doit normalement réunir l'ensemble de ces points. Il a obtenu, le cas échéant, son inscription sans problème.

Pour un Maître de conférences Docteur d'Etat, la Commission s'est assurée qu'il satisfaisait aux deux derniers critères.

Le problème a été incontestablement plus difficile pour les candidats sortant de ces profils habituels, Chargés de recherche au CNRS ou à l'INSERM, et surtout candidats relevant du 2ème concours.

Les principaux motifs de rejet ont été :

- absence d'encadrement de jeunes chercheurs,
- mise au ralenti ou arrêt des activités de recherche,
- expérience d'enseignement non significative (quelques heures annuelles d'enseignement dans un DEA sur un sujet pointu en relation directe avec l'activité de recherche n'ont pas été suffisantes pour les CR1),
- expérience professionnelle antérieure ne permettant pas de s'assurer que le candidat pourra remplir les missions habituelles d'un Professeur (pour le 2ème concours),
- inadéquation des activités de recherche et d'enseignement avec la Section 61.

QUELQUES CONSEILS A FAIRE PASSER AUX FUTURS CANDIDATS -

- Pour les Maîtres de conférences : ne pas cesser ses activités de recherche,
- Pour les Chargés de recherche : montrer son intérêt pour l'enseignement en effectuant quelques dizaines d'heures annuelles dans une filière EEA autre qu'un DEA,
- Pour les ingénieurs de l'industrie candidatant au titre du 2ème concours et ayant une activité de recherche industrielle voisine de celle d'un universitaire (publications dans les mêmes revues et les mêmes congrès) : passer l'Habilitation et montré son intérêt pour l'enseignement en effectuant quelques dizaines d'heures annuelles dans une filière EEA.

CONSEILS POUR LA CONSTITUTION DU DOSSIER -

Penser aux rapporteurs qui doivent se faire une opinion à partir du seul dossier. Fournir des indications précises sur la nature et le volume des enseignements effectués, étayées par des appréciations motivées des responsables d'enseignement. Classer les publications (revues avec comité de lecture, congrès avec actes, ouvrages, autres).

CONCLUSION -

La Commission a su éviter tout sectarisme, comme pour l'inscription sur la liste d'aptitude aux fonctions de Maître de conférences. En cas d'hésitation sur l'appartenance à la Section, le doute a toujours profité au candidat.

61ème section du CNU : Génie informatique, automatique et traitement du signal

COMPTE RENDU DE LA SESSION DE MAI 1992 INSCRIPTION SUR LA LISTE DE QUALIFICATION AUX FONCTIONS DE MAITRE DE CONFERENCES

QUELQUES CHIFFRES -

451 candidats.
244 inscrits sur la liste de qualification.
207 non inscrits dont 37 pour des raisons administratives (thèse non soutenue par exemple).

ÉLÉMENTS QUALITATIFS -

L'objectif final étant le recrutement de maîtes de conférences, le travail de la commission était d'évaluer les aptitudes à l'enseignement et à la recherche dans le champ d'activité couvert par la section 61 (identifié par ses mots clés : Génie Informatique, Automatique, Traitement du Signal).

Un candidat ayant passé une thèse, avec quelques publications (articles ou congrès sérieux ou actes) et ayant effectué des vacations dans l'enseignement supérieur (de l'ordre de 1/4 de service par an environ) a été inscrit sans problème.

Les principaux motifs de rejet sont :
- expérience de l'enseignement supérieur non significative,
- activité de recherche ne donnant pas lieu à publication,
- inadéquation des travaux de recherche et d'enseignement avec la section 61.

Il faut noter que, pour chacun de ces points, la Commission a cherché à tenir compte des situations particulières telles que : travaux confidentiels interdisant toute publication, thèse sous convention CIFRE ne permettant pas d'activités d'enseignement, etc.

QUELQUES CONSEILS A FAIRE PASSER AUX FUTURS CANDIDATS -

- montrer leur intérêt pour l'enseignement supérieur par une participation active,
- ne pas décrocher de la recherche après la soutenance de leur thèse.

POUR LA CONSTITUTION DU DOSSIER -

Ne pas oublier de fournir le rapport de soutenance de thèse, ainsi que les rapports préalables à la soutenance.

Donner des indications précises sur la situation professionnelle actuelle.

Donner des indications qualitatives et quantitatives sur l'expérience d'enseignement (disciplines, filière d'enseignement, volume horaire) si possible étayées par des appréciations des responsables d'enseignement.

CONCLUSION -

D'une manière générale, le sentiment des participants est que la Commission a su éviter les pièges de l'élitisme forcené et du sectarisme, tout en essayant de s'assurer de la bonne adaptation des candidats inscrits aux tâches qu'ils seraient amenés à exercer dans les différents types d'établissement.

Ph. BOLON - LAMII/FAST - BP 806 - 74016 ANNECY Cedex - GT8
P. BONTON- LE/CUST - 63174 AUBIERE Cedex - GT6
M. CORRAZZA - LASTI/ENSSAT - BP 447 - 22305 LANNION Cedex - GT8
G. DEMOMENT - L2S/ESE - Plateau du Moulon- 91190 GIF SUR YVETTE - GT5 GT8
I. MAGNIN - LTSU/INSA - Bât. 502 - 69621 VILLEURBANNE Cedex - GT5 GT8
V. LATTUATI - Labo. d'Automatique - ENSAM - 21 rue Pinel - 75013 PARIS - GT8
R. REYNAUD - IEF/PARIS 11 - Bât. 220 - 91045 ORSAY Cedex - GT3 GT8

PROMOTIONS AU TITRE DE L'ANNEE 1992

Les candidats à une promotion relevaient cette année de l'une des trois voies suivantes :

VOIE 1 - Procédure normale en deux temps : *phase locale* relevant des établissements qui pour chacun des cinq niveaux de promotion se sont vu attribuer un contingent toutes disciplines confondues, *phase nationale* ressortissant à la compétence de la commission n° 2 des sections du CNU.

VOIE 2 - Procédure d'avancement normale pour les enseignants-chercheurs relevant d'un "petit" établissement (moins de 500 enseignants-chercheurs ou moins de 30 professeurs). Il n'y a pas de phase locale, tout le contingent étant reporté sur la phase nationale.

VOIE 3 - Procédure d'avancement spécifique qui est de la compétence de la réunion des bureaux des commissions 2 des sections du groupe. Rappelons à ce propos que notre groupe, le groupe V, se compose des sections 25 (mathématiques pures), 26 (mathématiques appliquées) et 27 (informatique).

Le nombre de promotions à chaque niveau est déterminé par l'application au nombre de PROMOUVABLES du pourcentage suivant :

accès à MC1	: 41,6 %
accès à MC-HC	: 8,3 %
accès à PR1	: 13,8 %
accès à CE1	: 5,3 %
accès à CE2	: 27,1 %

Plus précisément, l'utilisation de ces coefficients magiques a été la suivante, pour chacune des trois voies.

Détermination du nombre de promotions dans la voie 1 -

Le CNU répartit la moitié des promotions calculée par application d'un coefficient magique à l'effectif de promouvables de la section. Le ministère n'a pas accepté qu'il y ait une compensation nationale tenant compte du choix des établissements (en fait il y a une légère compensation par le sens des arrondis). Ainsi, pour le pasage en HC des MC, il y avait 131 promouvables en 27° section. Le calcul du nombre de promotions en phase nationale est donc : 131 * 0,083 * 0,5 = 5,44. Notre commission a donc reçu un contingent de 5 promotions ce qui est correct compte tenu de ce que 6 collègues avaient été promus en juin par leur établissement.

Détermination du nombre de promotions dans la voie 2 -

La section a eu à sa disposition un contingent déterminé par application du ocefficient magique au nombre de promouvables de la section. Ces contingents sont très faibles car moins de 5 % des enseignants chercheurs relèvent de cette voie. Ainsi pour le passage la HC des MC, il y avait un seul promouvable (ouvrant droit à 0,083 promotion, contingent ramené à 0). En revanche pour l'accès à MC1, les promouvables étaient au nombre de 6 ce qui laisse espérer 6 * 0,416 = 2,496 promotions. La Commission a classé trois candidats. Le ministère a donné son accord ferme pour deux promotions (les bénéficiaires apparaissent sur les listes ci-après) et envisage favorablement notre demande d'une troisième promotion. Toutefois l'attribution définitive ne nous a pas été notifiée au moment où ces lignes sont écrites (le président de la commission préviendra, le cas échéant, le candidat classé troisième).

Détermination du nombre de promotions dans la voie 3 -

Le nombre de promotions est déterminé au niveau du groupe. Ainsi 196 MC étant promouvables à la HC par la voie 3 dans le groupe 5, ce groupe pouvait prétendre à 196 * 0,083 = 16,3 promotions à ce niveau et en a effectivement obtenu 16. Les bureaux ont réparti ces promotions au prorata des promouvables de chaque section, soit 5,6 et 5 promotions pour les sections 25, 26 et 27, respectivement.

Le lecteur trouvera ci-joint trois tableaux résumant les données et les résultats de ces calculs. En ce qui concerne le traitement "local" des informations relevant de la voie 1, les résultats sont en demi-teinte :

accès à MC1	: déficit de 9, partiellement rattrapé par la dotation nationale
accès à MC-HC	: rien à dire
accès à PR1	: rien à dire
accès à CE1	: déficit de 2, ramené à 1 par un rattrapage partiel au niveau national.
accès à CE2	: rien à dire.

On peut également s'intéresser à nos collègues matheux. La 25° section a été mal servie localement pour les passages en MC-HC et PR1, mais fort appréciée des autres disciplines pour l'accès à la CE1.

Remarquons qu'il est peu astucieux de renoncer à l'avancement spécifique (voie 3) si de ce fait on retombe dans la voie 2 parce que l'on appartient à un "petit" établissement.

Un point inquiétant est l'avenir des promotions à la Hors Classe des Maîtres de Conférences. L'année 1992 est la troisième et dernière année du plan mis en place dans le cadre de la revalorisation de la condition des enseignants. A partir de l'an prochain le nombre de possibilités de promotions sera très inférieur à celui de cette année qui est globalement du même ordre de grandeur que celui des deux années précédentes. Lors de la réunion de groupe une motion a été adoptée et adressée au ministre (voir annexe ci-dessous).

MOTION DU 5° GROUPE DU C.N.U.

Au cours des réunions des commissions des sections 25, 26, 27 et du 5° groupe, de nombreux excellents candidats à la Hors Classe des Maîtres de Conférences équivalents aux candidats retenus n'ont pu bénéficier de la promotion à la Hors Classe à cause du trop faible nombre de places disponibles ; ceci tant pour l'avancement normal (voies n° 1 et 2) que pour l'avancement dit spécifique (voie n° 3).

Par souci d'équité, afin de résorber ces nombreuses candidatures de valeur et d'assurer le fonctionnement de qualité des universités, il apparaît opportun de poursuivre la politique de création d'emplois de MCF Hors Classe avec une ampleur au moins égale à celle de ces dernières années.

En outre, cette mesure maintiendrait le re-pyramidage récent, compte tenu de l'augmentation du nombre des MCF.

L Illusie

Luc ILLUSIE

Président section 25 Com. 2

Jacques LENFANT

Président section 27 Com. 2.

Jean Claude Nédélec

Président 26e com. 2

PANORAMA DU GROUPE 5

1. Procédure normale - hors petits établissements (VOIE 1) -

Accès à	25° section				26° section				27° section			
	A	B	C	D	A	B	C	D	A	B	C	D
MC1	54	22,5	13	9	66	27,5	10	15	152	63,2	23	34
MC HC	271	22,5	7	12	196	16,3	6	8	131	10,9	6	5
PR1	195	26,9	9	14	184	25,4	14	12	179	24,7	12	13
CE1	179	9,5	7	4	132	7,0	4	3	114	6,0	1	4
CE2	22	6,0	2	3	15	4,1	-	2	10	2,7	1	2

Légende :

A : promouvables
B : application des coefficients magiques aux nombres de promouvables
C : nombre de collègues promus par leur établissement
D : promotions qui seront attribuées par le CNU

2. Procédure normale - petits établissements (VOIE 2) -

Accès à	25° section			26° section			27° section		
	A	B	D	A	B	D	A	B	D
MC1				7	2,9	3	6	2,5	2 ou 2
MC HC				1	0,1	0	1	0,1	0
PR1	12	1,7		25	3,45	3	10	1,4	1
CE1	4	0,2		3	0,2	0	5	0,3	0
CE2				-			-		0

3. Procédure spécifique (VOIE 3) -

Accès à	Groupe			25° section			26° section			27° section		
	A	B	D	A	E	F	A	E	F	A	E	F
MC1	47	19,6	20	7	2,98	3	1	5,11	5	28	11,9	12
MCHC	196	16,3	16	59	4,82	5	73	5,96	6	64	5,22	5
PR1	59	8,1	8	16	2,17	2	21	2,85	3	22	2,98	3
CE1	45	2,4	2	10	0,44	0	17	0,76	1	18	0,80	1
CE2	6	1,6	2	3	1	1	0	0	0	3	1	1

Légende :

A, B, D : comme précédemment
E : répartition (théorique) du contingent du groupe au prorata des promouvables
F : répartition effective.

Promotions attribuées par les établissements (VOIE 1, phase locale)

Passage en MC1 (23)

VIDAL Didier	Paris 6
ALLYS Loïc	Le Mans
BERGERE Michel	Orléans
BRYGOO Anne-Marie ép. NAULLEAU	Paris 6
BIANCHI Giuliana	Bordeaux I
TANGUY Roger	Paris 8
PETER Philippe	Nantes
BOSSUT Francis	Lille 1
AUTHOSSERRE Annie ép. CAVARERO	Nice
BORNE Isabelle	Paris 5
CARRIER Fabienne ép. LAGNIER	Grenoble 1
DAIGREMONT Simone ép. PIMONT	Lyon 1
DEFUDE Bruno	Ensimag
FERNANDEZ Jean-Claude	Grenoble 1
GALLINARI Patrick	Paris 11
LESVENTES Gilles	Rennes I
MONTANVERT Annick	Grenoble 2
LUGIEZ Denis	Nancy I
OU HALIMA Mohamed	INSA Lyon
OURIACHI Khadir	Valenciennes
SEEBOLD Patrice	Paris 7
TRIGANO Philippe	Compiègne

Passage en MC HC (6)

GIANNESINI François	Aix-Marseille 2
DELDICQ André	Paris 7
FLAVIGNY Bruno	Paris 6
MANARA Jean-Henry	Paris 7
HERVIER Yves	Nice
PROUST Christian	Tours

Passage en PR1 (12)

PECCOUD François	Grenoble 2
MORGENSTERN Jacques	Nice
RAYNAUD Yves	Toulouse 3
POMEROL Jean-Charles	Paris 6
BAZEX Pierre	Toulouse 3
LORETTE Guy	Rennes 1
NICOLLE Anne ép. ADAM	Caen
BASSANO Jean-Claude	Orléans
CUNIN Pierre	Grenoble 1
PANSIOT Jean-Jacques	Strasbourg 1
KRIEF Jacqueline ép. VAUZEILLES	Paris 13
CORMIER Laurence ép. PUEL	Paris 11

Passage en CE1 (1)

LAZARD Daniel	Paris 6

Passage en CE2 (1)

DELOBEL Claude	Paris 11

Promotions attribuées par la Section 27
(VOIE 1, phase nationale)

Passage en MC1 (promotion au 1.1.92)

Fouad AYOUB BADRAN	CNAM Paris
Jean-Paul BAHSOUN	Toulouse 3
Kamel BARKAOUI	CNAM Paris
Marie-Pierre BEAL	Paris 7
Béatrice BERARD	ENS Cachan
Corine CAUVET	Paris I
Michel DAYDE	INP Toulouse
Jacques DIGOUT	Toulouse 3 IUT A
Marie-Christine FAUVET	Grenoble 1
Odile GAUTRAND	Toulouse 2 IUT B
Philippe GENOUD	Grenoble 1
Claire HANEN	Paris 6
Jean-Louis IMBERT	Aix-Marseille 2
Georges KHALIL	Reims
Christophe LECERF	La Réunion
Franck MARCHETTI	Metz IUT Metz
Philippe MAURAN	INP Toulouse
Jean-François MEHAUT	Lille 1
Habib MEHREZ	Paris 6
Jean-Jacques MILAN	Lyon 1
Noureddine MOUADDIB	Nancy 1
Hassan MOUNTASSIR	Besançon
Marc NEVEU	Dijon
Odile PAPINI	Aix-Marseille 2 IUT Aix
Jean-Pierre REGOURD	Nice
Bruno WARIN	Lille 1 IUT Calais
Karine ZAMPIERI	Haute-Alsace
François Xavier TESTARD-VAILLANT	ENS St Cloud

Passage en MC1 (promotion au 10.1.92)

Francis ALEXANDRE	Nancy 1
Michel FUTTERSACK	Le Mans
Jean-Louis ROCH	Ensimag Grenoble
Philippe VIDAL	Toulouse 3 IUT A
Jean-Marc VINCENT	Grenoble 1
Danielle ZIEBELIN	Grenoble 1

Passage en MC-HC (promotion au 1.1.92)

Yves KERGALL	Avignon
Monique ROLLEY ép. LE COZ	Brest

Passage en MC-HC (promotion au 1.10.92)

André COUVERT	Rennes 1
Dominique OLLIVIER-PALLUD	Paris 7
André SCHAFF	Nancy 1

Passage en PR1

Dominique BORRIONE	Grenoble
André BOUCHET	Le Mans
Jean-Pierre CABANEL	INP Toulouse
Gérard FERRAND	Orléans
Gérard JACOB	Lille 1
William JALBY	Versailles
Brigitte PLATEAU	INP Grenoble
Alain QUILLIOT	Clermont 2
Jean-Claude RAOULT	Rennes 1
Yves ROBERT	Lyon 1
René SCHOTT	Nancy 1
Marie-Claude TCHERKEZ ép. HEYDEMANN	Paris II IUT Orsay
Bernard VAUQUELIN	Bordeaux 1

Passage en CE1

Michel CHEIN	Montpellier 2
Bruno COURCELLE	Bordeaux I
Marie-Claude GAUDEL	Paris 11
Gérard VEILLON	INP Grenoble

Passage en CE2

Jean-Claude BOUSSARD	Nice
Robert CORI	Bordeaux 1

Promotions attribuées par la Section 27 (VOIE 2)

Passage en MC1

Jean-Luc COQUIDE	IUFM Lille
Christine PORQUET	ISMRA Caen

Passage en PRI

Jean-Michel PECUCHET	INSA Rouen

Informaticiens promus par le groupe (VOIE 3)

Passage en MC1 (12)

Bernard COULETTE	INP Toulouse
Sylvie VOLDMAN ép. SZULMAN	Paris 13
Azim ROUSSANALY	Nancy 2
Marina PIERCEY ép. MARMONIER	Compiègne
Odile CARRIERE ép. MOLINA-LIRA	Pau
Amal SAYAH	Toulouse 3
Félix PAOLETTI	Paris 6
Gabriel MICHEL	Metz
Marc DALMAU	Pau
Anne-Marie BLATTNER ép. DILL	INSA Lyon
Daniel CHAMBON	Toulouse 3
Martine GAUTIER ép. CAMONIN (au 1.10.92)	Nancy 1

Passage en MC-HC (5)

au 1.1.92

Anne-Marie BLANC ép. ALQUIER	Toulouse 1
Jacques BERNARD	Clermont 1

au 1.10.92

Régine LAUDET ép. RAYNAUD	Toulouse 3
Martine ROUSSEAU	PARIS 11
Jean-Louis NEBUT	Rennes 1

Passage en PR1 (3)

Guy ROMIER	Grenoble 2
Catherine BEAUDELOT ép. ROUCAIROL	Paris 6
Odile LEGLISE ép. THIERY	Nancy 2

Passage en CE1 (1)

Jean-Pierre FINANCE	Nancy 1

Passage en C2 (1)

Dominique PERRIN	Paris 7

RECRUTEMENTS 92

Nous publions ci-après la liste provisoire des candidats retenus par les CSE, en 1992, et qui sont nommés ou en cours de nomination. Les modalités de recrutement ayant été modifiées, la constitution de cette liste est désormais une tâche plus difficile que par le passé. Nous tenons à remercier les services compétents du MEN qui nous ont apporté leur concours. Bien entendu, cette liste est publiée avec les réserves d'usage, sous la responsabilité de SPECIF, et n'engage nullement le MEN.

SPECIF

RECRUTEMENTS PROFESSEURS

CLASSÉS PAR CANDIDATS

ETABLISSEMENT	NOM
INSA Rouen	ABDULRAB Habib
Paris 6	ACCART Thérèse, ép. HARDIN
Besançon	AGGARWAL Nan Lal
Lyon 3	AYMARD Danielle, ép. BOULANGER
Paris 11	BEAUDOIN LAFON Michel
Paris 13	BIDOIT Nicole
Versailles- St Quentin	BOUZEGHOUB Mokrane
Rouen	CHAMPARNAUD Jean-Marc
Rennes 1	COUSIN Bernard
Toulouse 2 (IUT-B)	CRAMPES Jean-Bernard
Paris 13 (IUT Villetaneuse)	DAGUE Philippe
Paris 8	DEGREMONT Jean-François
Lille 1	DEKEYSER Jean-Luc
Le Mans	DELEGLISE Paul
C.N.A.M.	DEWEZ Louis
Toulouse 3 (IUT-A)	DOURS Daniel
Paris 6	ESTRAILLER Pascal
Rennes 2 (IUT Vannes)	FRISON Patrice
Paris 6	GALLINARI Patrick
Grenoble 2	GIRAUDIN Jean-Pierre
Lille 1 (IUT-A)	GRIMONPREZ Georges
Bordeaux 1	GUITTON Pascal
I.N.P. Grenoble	JACQUET Paul
Evry-Val d'Essonne	KONIG Jean-Claude
Nice (IUT Nice)	LE THANH Nhan
Toulouse 3	MASSIP PAILHES Louis
Montpellier 2	MAURER Marie, ép. VILAREM
Pau	OURIACHI Khadir
Paris 9	PASCHOS Evagelos
Paris 12	PELZ Elisabeth
Toulouse 3 (IUT-A)	PERCEBOIS Christian
Limoges	PLEMENOS Dimitrios
Strasbourg 1	RONSE Christian
Nice	RUEHER Michel
Paris 13 (IUT Villetaneuse)	SCHWER Sylviane
Amiens	SEEBOLD Patrice
Poitiers	STOYKOVA Lozka, ép. POPOVA
Grenoble 1	THEIS Marie-Françoise, ép. BRUANDET
I.P. Sévenans	WENDLING Serge
Bordeaux 1	ZVONKINE Alexandre

RECRUTEMENTS PROFESSEURS

CLASSÉS PAR ÉTABLISSEMENTS

ETABLISSEMENT	NOM
Amiens	SEEBOLD Patrice
Besançon	AGGARWAL Nan Lal
Bordeaux 1	GUITTON Pascal
Bordeaux 1	ZVONKINE Alexandre
C.N.A.M.	DEWEZ Louis
Evry-Val d'Essonne	KONIG Jean-Claude
Grenoble 1	THEIS Marie-Françoise, ép. BRUANDET
Grenoble 2	GIRAUDIN Jean-Pierre
I.N.P. Grenoble	JACQUET Paul
I.P. Sévenans	WENDLING Serge
INSA Rouen	ABDULRAB Habib
Le Mans	DELEGLISE Paul
Lille 1	DEKEYSER Jean-Luc
Lille 1 (IUT-A)	GRIMONPREZ Georges
Limoges	PLEMENOS Dimitrios
Lyon 3	AYMARD Danielle, ép. BOULANGER
Montpellier 2	MAURER Marie, ép. VILAREM
Nice	RUEHER Michel
Nice (IUT Nice)	LE THANH Nhan
Paris 6	ACCART Thérèse, ép. HARDIN
Paris 6	ESTRAILLER Pascal
Paris 6	GALLINARI Patrick
Paris 8	DEGREMONT Jean-François
Paris 9	PASCHOS Evagelos
Paris 11	BEAUDOIN LAFON Michel
Paris 12	PELZ Elisabeth
Paris 13	BIDOIT Nicole
Paris 13 (IUT Villetaneuse)	DAGUE Philippe
Paris 13 (IUT Villetaneuse)	SCHWER Sylviane
Pau	OURIACHI Khadir
Poitiers	STOYKOVA Lozka, ép. POPOVA
Rennes 1	COUSIN Bernard
Rennes 2 (IUT Vannes)	FRISON Patrick
Rouen	CHAMPARNAUD Jean-Marc
Strasbourg 1	RONSE Christian
Toulouse 2 (IUT-B)	CRAMPES Jean-Bernard
Toulouse 3	MASSIP PAILHES Louis
Toulouse 3 (IUT-A)	DOURS Daniel
Toulouse 3 (IUT-A)	PERCEBOIS Christian
Versailles- St Quentin	BOUZEGHOUB Mokrane

27ème section Informatique Concours 1992

RECRUTEMENTS MAITRES DE CONFERENCES

CLASSÉS PAR CANDIDATS

ETABLISSEMENT	NOM
Rennes 2 (IUT Vannes)	ADAM Michel
INP Toulouse	AMESTOY Patrick
Orléans	ANDRIANARIVELO Rakotoarimanana
CNAM	APONTE-GARCIA Maria
ENS Lyon	AUDEBAUT Christian
Paris 8	BAFFOY Thierry
Paris 12	BARI Mohammed
Toulouse 1	BASTIDE Rémi
INSA Lyon	BENHARKAT Aïcha Nabila
Cergy-Pontoise (IUT Cergy)	BENNACEUR Hachemi
Versailles St-Quentin (IUT Velizy)	BENNIS Karine, ép. ZEITOUNI
Nantes (IRESTE)	BENOIS Jenny
Dijon (IUT Dijon)	BENSLIMANE Djamal
Pau	BESSAGNET Marie-Noëlle
CNAM	BONGARD Christine ép. CROCHEPEYRE
Lyon 1 (IUT-A)	BONNET CHristine
Nice (IUT Nice)	BORELLI Evelyne, ép. VITTORI
Evry-Val d'Essonne	BOUABDALLAH Abdelmadjid
Le Havre (IUT Le Havre)	BOUDEBOUS Dalila
U.T. Compiègne	BOUFFLET Jean-Paul
INSA Lyon	BOULICAUT Jean-François
Tours	BOUQUARD Jean-Louis
Valenciennes	BOURZOUFI Abdelhafid
Lyon 3	BOUZIDI Laïd
Toulouse 3	BRAS Myriam
Paris 13 (IUT Villetaneuse)	CALMEN Françoise, ép. GAYRAL
Rouen	CARRE Christophe
Paris 6	CHAILLOUX Emmanuel
Besançon	CHANUDET Sylvie, ép. DAMY
Bordeaux 1	CHAUMETTE Serge
Paris 12	CHESNOT Joëlle, ép. COHEN
Aix Marseille 2 (IUT Aix)	CICCHETTI Rosine
Caen	CLERIN Françoise, ép. DEBART
Nice (IUT Nice)	COLLARD Philippe
Nice	COLLAVIZZA Hélène
Nancy 1	COLLIN Suzanne
Lille 1 (IUT Calais-Dunkerque)	COPPIN Bénédicte, ép. TALON
CNAM	CUBAUD Pierre
Bordeaux 3	DAI Mo
Versailles-St Quentin	DAVID Pierre

Nancy 1	DEGLI Jocelyne, ép. ROUYER
Le Mans	DELOZANNE Elisabetch
Montpellier 2	DONY Christophe
Toujouse 3	DUGAT Vincent
Limoges (IUT Limoges)	DUMONT Jacques
Paris 6	DUTHEILLET LAMONTHEZIE Claude
Angers	DUVAL Béatrice
Caen	EL MOATAZ Billah Abderrahim
Mulhouse	ELBAZ Mounir
Paris 8	FEAT Jym
Nice	FEDELE Carine
Chambéry	FERRARIS Christine
Versailles-St Quentin	FINANCE Béatrice
Marne la Vallée	FRUITET Jean
Amiens (IUT Amiens)	GAY Valérie
Paris 11	GIBET Sylvie
Grenoble 1	GIROD Xavier
Aix Marseille 2	GRANDCOLAS Stéphane
Paris 13 (IUT St Denis)	GRANDEMANGE Philippe
Bordeaux 1	GRIFFAULT Alain
Paris 1	GROSZ Georges
Amiens	GUERIN Jean-Luc
Corte (IUT Corte)	HATIMI Mostafa
Reims	HEBBACHE Abdelhamid
Amiens	HERNANDEZ PEREZ Marisela
INP Toulouse (ENSEEIHT)	HOUZET Dominique
Montpellier 2	HUCHARD Marianne
Besançon (IUT Belfort)	JACQUES Isabelle
Nantes (IUT Nantes)	JACQUIN Christine
Lille 1	JANOT Stéphane
IUFM Versailles	KAZEMI Nejad Amir
Nantes (IUT Nantes)	KHAMMACI Tahar
Nice	KOUNALIS Emmanuel
Toulon	LANGEVIN Philippe
Versailles-St Quentin (IUT Vélizy)	LARCHEVEQUE Jean-Marie
Toulouse 1 (IUT Rodez)	LBATH Redouana
Bordeaux 2	LE BLANC Benoît
Rouen	LECROQ Thierry
Limoges (IUT Limoges)	LELASSEUX Philippe
Clermont Ferrand 2	LI JIAN Jin
Toulouse 3	MAGNAUD Patrick
La Rochelle	MAHMOUD Mohamed Youssri
Chambéry (IUT Annecy)	MAIRE Jean-Luc
Rennes 1	MAISEL Eric
Paris 7	MANTACI Roberto
Lille 1	MARQUET Philippe
Paris 13	MASSERON Marcel
Nantes	MAURAS Christophe
Nantes	MEKADAUCHE Abdelouahab
Metz	MICHEL Dominique
INSA Rennes	MOLNAR Miklos
Aix Marseille 2	MONDAIN MONVAL Pierre
Versailles-St Quentin	MORIZE Isabelle, ép. QUILIO
ENS Lyon	MORVAN Michel
Toulon	MURISASCO Elisabeth
Metz (IUT Metz)	NADIF Mohamed
Le Havre (IUT Le Havre)	NAKECHBANDI Moustafa
Avignon	NOCERA Pascal
Grenoble 2	OQUENDO SIMOES Flavio
Rennes 1 (IUT Lannion)	OUALA Allah Mohamed
Limoges	OULD Braham Tayeb

ENS Lyon	PAUGAM Hélène, ép. MOISY
Pau (IUT Pau-Bayonne)	PECATTE Jean-Marie
Paris 13 (IUT Villetaneuse)	PEKERGIN Mehmet
Paris 6	PERNY Patrice
Lyon 1	PERRAUD Florence, ép. ECHALIER
IUFM Dijon	PERROT Elisabeth, ép. GAVIGNET
Lille 1	PETITOT Michel
Valenciennes	PIECHOWIAK Sylvain
Rennes 1 (ENSSAT Lannion)	PIVERT Olivier
Valenciennes	POIRRIEZ Vincent
Rennes 2 (IUT Vannes)	PORTEJOIE Philippe
Brest	POTTIER Bernard
Montpellier 2 (IUT Montepllier)	REITZ Philippe
Versailles-St Quentin (IUT Vélizy)	ROBBA Isabelle
Versailles-St Quentin	ROQUES Michelle, ép. CLAUDE
Le Havre	SADEG Brahim
Caen	SAQUET Jean-Paul
Lyon 3	SAUVAGE Caroline, ép. WINTERGERST
Aix Marseille 3	SEBBAN Colette, ép. FAUCHER
Littoral	SLOWINSKI Karine
Nancy 2	SMAILI Kamel
Nancy 1 (IUT-B)	SONG YE Qiong
Paris 5	SOTO Michel
Tours	TAGHELIT Mohamed
Strasbourg 1	TAJINE Mohamed
Tours	TREPIED Claude
Rennes 1	VIHO GBEDE Gagnon
Rennes 1	WORC'H Raoul
Grenoble 1	WAILLE Philippe
Paris 11 (IUT Sceaux)	ZARATE Pascale
Paris 5 (IUT av. de Versailles)	ZIANE Mikal
Marne la Vallée	ZIPSTEIN Marc
Paris 11	ZISSIMOPOULOS Vassilis

RECRUTEMENTS MAITRES DE CONFERENCES

CLASSÉS PAR ÉTABLISSEMENTS

ETABLISSEMENT	NOM
Aix Marseille 2 (IUT Aix)	CICCHETTI Rosine
Aix Marseille 2	GRANDCOLAS Stéphane
Aix Marseille 2	MONDAIN MONVAL Pierre
Aix Marseille 3	SEBBAN Colette, ép. FAUCHER
Amiens (IUT Amiens)	GAY Valérie
Amiens	GUERIN Jean-Luc
Amiens	HERNANDEZ PEREZ Marisela
Angers	DUVAL Béatrice
Avignon	NOCERA Pascal
Besançon	CHANUDET Sylvie, ép. DAMY
Besançon (IUT Belfort)	JACQUES Isabelle
Bordeaux 1	CHAUMETTE Serge
Bordeaux 1	GRIFFAULT Alain
Bordeaux 2	LE BLANC Benoît
Bordeaux 3	DAI Mo
Brest	POTTIER Bernard
Caen	CLERIN Françoise, ép. DEBART
Caen	EL MOATAZ Billah Abderrahim
Caen	SAQUET Jean-Paul
Cergy-Pontoise (IUT Cergy)	BENNACEUR Hachemi
Chambéry	FERRARIS Christine
Chambéry (IUT Annecy)	MAIRE Jean-Luc
Clermont Ferrand 2	LI JIAN Jin
CNAM	APONTE-GARCIA Maria
CNAM	BONGARD Christine ép. CROCHEPEYRE
CNAM	CUBAUD Pierre
Corte (IUT Corte)	HATIMI Mostafa
Dijon (IUT Dijon)	BENSLIMANE Djamal
ENS Lyon	AUDEBAUT Christian
ENS Lyon	MORVAN Michel
ENS Lyon	PAUGAM Hélène, ép. MOISY
Evry-Val d'Essonne	BOUABDALLAH Abdelmadjid
Grenoble 1	GIROD Xavier
Grenoble 1	WAILLE Philippe
Grenoble 2	OQUENDO SIMOES Flavio
INP Toulouse	AMESTOY Patrick
INP Toulouse (ENSEEIHT)	HOUZET Dominique
INSA Lyon	BENHARKAT Aïcha Nabila
INSA Lyon	BOULICAUT Jean-François
INSA Rennes	MOLNAR Miklos
IUFM Dijon	PERROT Elisabeth, ép. GAVIGNET
IUFM Versailles	KAZEMI Nejad Amir
La Rochelle	MAHMOUD Mohamed Youssri

Le Havre (IUT Le Havre)	BOUDEBOUS Dalila
Le Havre (IUT Le Havre)	NAKECHBANDI Moustafa
Le Havre	SADEG Brahim
Le Mans	DELOZANNE Elisabetch
Lille 1 (IUT Calais-Dunkerque)	COPPIN Bénédicte, ép. TALON
Lille 1	JANOT Stéphane
Lille 1	MARQUET Philippe
Lille 1	PETITOT Michel
Limoges (IUT Limoges)	DUMONT Jacques
Limoges (IUT Limoges)	LELASSEUX Philippe
Limoges	OULD Braham Tayeb
Littoral	SLOWINSKI Karine
Lyon 1 (IUT-A)	BONNET CHristine
Lyon 1	PERRAUD Florence, ép. ECHALIER
Lyon 3	BOUZIDI Laïd
Lyon 3	SAUVAGE Caroline, ép. WINTERGERST
Marne la Vallée	FRUITET Jean
Marne la Vallée	ZIPSTEIN Marc
Metz	MICHEL Dominique
Metz (IUT Metz)	NADIF Mohamed
Montpellier 2	DONY Christophe
Montpellier 2	HUCHARD Marianne
Montpellier 2 (IUT Montepllier)	REITZ Philippe
Mulhouse	ELBAZ Mounir
Nancy 1	COLLIN Suzanne
Nancy 1	DEGLI Jocelyne, ép. ROUYER
Nancy 1 (IUT-B)	SONG YE Qiong
Nancy 2	SMAILI Kamel
Nantes (IRESTE)	BENOIS Jenny
Nantes (IUT Nantes)	JACQUIN Christine
Nantes (IUT Nantes)	KHAMMACI Tahar
Nantes	MAURAS Christophe
Nantes	MEKADAUCHE Abdelouahab
Nice (IUT Nice)	BORELLI Evelyne, ép. VITTORI
Nice (IUT Nice)	COLLARD Philippe
Nice	COLLAVIZZA Hélène
Nice	FEDELE Carine
Nice	KOUNALIS Emmanuel
Orléans	ANDRIANARIVELO Rakotoarimanana
Paris 1	GROSZ Georges
Paris 5	SOTO Michel
Paris 5 (IUT av. de Versailles)	ZIANE Mikal
Paris 6	CHAILLOUX Emmanuel
Paris 6	DUTHEILLET LAMONTHEZIE Claude
Paris 6	PERNY Patrice
Paris 7	MANTACI Roberto
Paris 8	BAFFOY Thierry
Paris 8	FEAT Jym
Paris 11	GIBET Sylvie
Paris 11 (IUT Sceaux)	ZARATE Pascale
Paris 11	ZISSIMOPOULOS Vassilis
Paris 12	BARI Mohammed
Paris 12	CHESNOT Joëlle, ép. COHEN
Paris 13 (IUT St Denis)	GRANDEMANGE Philippe
Paris 13 (IUT Villetaneuse)	CALMEN Françoise, ép. GAYRAL
Paris 13	MASSERON Marcel
Paris 13 (IUT Villetaneuse)	PEKERGIN Mehmet
Pau	BESSAGNET Marie-Noëlle
Pau (IUT Pau-Bayonne)	PECATTE Jean-Marie
Reims	HEBBACHE Abdelhamid
Rennes 1	MAISEL Eric

Rennes 1 (ENSSAT Lannion)	PIVERT Olivier
Rennes 1 (IUT Lannion)	OUALA Allah Mohamed
Rennes 1	VIHO GBEDE Gagnon
Rennes 1	WORC'H Raoul
Rennes 2 (IUT Vannes)	ADAM Michel
Rennes 2 (IUT Vannes)	PORTEJOIE Philippe
Rouen	CARRE Christophe
Rouen	LECROQ Thierry
Strasbourg 1	TAJINE Mohamed
Toujouse 3	DUGAT Vincent
Toulon	LANGEVIN Philippe
Toulon	MURISASCO Elisabeth
Toulouse 1	BASTIDE Rémi
Toulouse 1 (IUT Rodez)	LBATH Redouana
Toulouse 3	BRAS Myriam
Toulouse 3	MAGNAUD Patrick
Tours	BOUQUARD Jean-Louis
Tours	TAGHELIT Mohamed
Tours	TREPIED Claude
U.T. Compiègne	BOUFFLET Jean-Paul
Valenciennes	BOURZOUFI Abdelhafid
Valenciennes	PIECHOWIAK Sylvain
Valenciennes	POIRRIEZ Vincent
Versailles-St Quentin	DAVID Pierre
Versailles-St Quentin	FINANCE Béatrice
Versailles-St Quentin (IUT Vélizy)	LARCHEVEQUE Jean-Marie
Versailles-St Quentin	MORIZE Isabelle, ép. QUILIO
Versailles-St Quentin (IUT Vélizy)	ROBBA Isabelle
Versailles-St Quentin	ROQUES Michelle, ép. CLAUDE
Versailles St-Quentin (IUT Velizy)	BENNIS Karine, ép. ZEITOUNI

RECRUTEMENTS PRAG

ETABLISSEMENT	NOM
IUT Quimper	BARBIER Marc
IUT La Rochelle	BERNARD Laurence
IUT Avignon	CARO Yves
IUT Montpellier-Nîmes	FAURE Jean-Paul
IUT Rennes 1	JOURDROUIN René
IUT Nantes	LAIME Véronique
ENS Fontenay Saint-Cloud	LEFEVRE Marie-Claire
Université Brest	PELLEN Raymonde
Université Lille 1	PEREAU Martine
IUT à Lyon 1	PILLOT Allain
IUT Metz	RIBAU Robert

SECTION 07 DU C.N.R.S.

. La Recherche en Informatique au CNRS

. Jurys d'admissibilité 1992

. Session d'Automne 1992

La Recherche en Informatique au CNRS

Les informations et considérations qui suivent n'engagent que leur auteur.

Le CNRS comprend 7 Département Scientifiques et 40 sections du Comité National. L'Informatique dépend du Département SPI (*Sciences Pour l'Ingénieur*) et de la section 07 *Sciences et technologies de l'information (informatique, automatique, traitement du signal)*. Le SPI représente un peu moins de 10% du CNRS.

Quelques chiffres-repères (approximatifs).

Effectifs:

	Informatique	section 07	SPI	Toutes disciplines
Chercheurs CNRS	190	290	1150	11500
Ens.-ch. ds Unité CNRS	900		2500	12000
Total Enseignants-cherch.	2000	2800	6600	48000
ITA CNRS	170-200	300	1100	12000 dans labos
ATOS MEN	180			
Unités CNRS	30	45	160	1350

Un (chimérique) labo moyen d'informatique lié au CNRS a donc 30 enseignants chercheurs, 6 Chercheurs CNRS, 6 ITA et 6 ATOS. Toutes disciplines confondues, on obtient 9 enseignants chercheurs, 8 chercheurs CNRS et 8 ITA (attention: les unités propres sont prises en compte dans ces calculs). En outre, presque un enseignant chercheur en informatique sur 2 est dans une unité CNRS, contre un sur quatre toutes disciplines confondues. Il y a un chercheur CNRS en informatique pour 10 universitaires en informatique, contre un sur 5 au total; mais il faut tenir compte de l'INRIA, qui a des effectifs comparables à ceux de la section 07 du CNRS.

En résumé, par rapport à l'ensemble des disciplines, un laboratoire d'informatique associé au CNRS se distingue surtout par le pourcentage élevé d'Universitaires qui y sont rattachés. Le rapport chercheurs CNRS / Universitaires est à peine moitié de la moyenne si on tient compte du seul CNRS, mais proche de la moyenne si on tient compte de l'INRIA. D'où l'importance d'une politique concertée avec cet institut.

Budget: Le personnel représente 75% du budget. Comme dans les autres départements, l'essentiel du budget hors salaires (165 MF environ) est récurrent. Le SPI souhaite dégager plus d'argent pour développer une politique incitative.

Les contrats: Les labos du SPI, et ceux d'informatique en particulier, sont ceux qui ont proportionnellement le plus de contrats. Un labo du SPI déclare en moyenne 2.3 MF de contrats (2.5 pour l'informatique).

La politique scientifique du SPI:

Jean-Jacques Gagnepain, Directeur du Département depuis début 92, redéfinit la politique du SPI en large concertation avec les communautés scientifiques. Des comités de réflexion thématiques viennent de rendre leurs conclusions. Les nouvelles OST seront en nombre restreint. En ce qui concerne la section 07, les titres (provisoires) sont

- Architecture des Systèmes informatiques
- Sciences du Logiciel
- Automatique, aide à la Décision
- Signaux et Images
- Intelligence Artificielle, Communication Hommes-Machines, Robots.

Des thèmes prioritaires sont en cours de définition.

Les priorités du département sont en outre:

- Relations inter-départements (opérations interdisciplinaires)
- Relations internationales (à développer en particulier avec l'Europe, l'Extrême-Orient, les Pays de l'Est, à maintenir avec l'Amérique du Nord); accueil de chercheurs; cofinancement de thèses, structures européennes (responsable: Jean-Claude André, Directeur Adjoint)
- Politique régionale (plans Etats-Régions)
- Formation, relations industrielles, communication.

En ce qui concerne l'Informatique:

- Les relations entre le CNRS et l'INRIA sont en train de se structurer. Le SPI y joue un rôle moteur (responsable: Bernard Dubuisson, Directeur Adjoint).
- L'interdisciplinarité, par l'intermédiaire des PIR ou à l'initiative du SPI.

****

La politique de l'emploi scientifique au CNRS:

La Direction de la Stratégie et des Programmes du CNRS a lancé en juin une réflexion sur ce thème. Voici quelques éléments:

Constats:

- Vieillissement du corps des chercheurs
- Nécessité de stabiliser le nombre de postes budgétaires
- Besoins de recrutements massifs dans l'enseignement supérieur
- Le SPI est atypique (l'info en particulier, plus jeune, avec ses récents recrutements)

Quelques chiffres:

- Actuellement, 250 départs par an, dont moins de 100 en retraite, pour 400 recrutements.
- Le flux annuel de départs spontanés est actuellement de 1% au CNRS. C'est au SPI qu'il est le plus élevé (1.7%), par départs dans le privé et surtout à l'Université.
- Un quart des chercheurs CNRS enseigne déjà à l'Université.

D'où les axes de réflexion suivants:

- Incitation au passage du CNRS vers l'Université (une partie seulement des chercheurs recrutés feraient toute leur carrière au CNRS, une forte majorité finissant alors DR). L'avantage pour l'Université serait une meilleur répartition de l'âge de ses personnels.
- Diminution du recrutement
- En échange, les postes ainsi libérés serviraient à des post-docs et à l'accueil, en particulier à l'accueil pour quelques années de nombreux jeunes (ou moins jeunes) universitaires. Ces mouvements pourraient faire l'objet d'une politique contractuelle avec les établissements et les Unités CNRS.
- Les départements pourraient avoir une politique différenciée.

Conclusion: Eléments à méditer.

L'interdisciplinarité et l'identité:

L'info est souvent perçue de l'extérieur plus comme un service que comme une discipline, mais elle est très sollicitée. A nous de nous affirmer comme discipline tout en gérant les multiples appels interdisciplinaires, qui sont un atout dans la politique scientifique du CNRS, organisme pluridisciplinaire par excellence.

La communauté des universitaires et des chercheurs:

Originalités: Nombreuse et jeune population universitaire dans les unités CNRS.

Deux organismes publics de recherche de taille comparable en informatique: le CNRS et l'INRIA.

La future politique de l'emploi scientifique au CNRS

Max Dauchet,
chargé de mission pour l'Informatique
au département SPI du CNRS

Résultats de Jury d'admissibilité 1992
Section 07

(RÉSULTATS DES CONCOURS CR2 ET CR1, DONNÉS SOUS TOUTE RÉSERVE)

706
Non Affiche
4 postes

Admissibilite :
1. Wang Dongming
2. Ghorbel Fathi
3. Laboissiere Rafael
4. Desbat Laurent
5. Zasadzinski Michel
6. Peladeau Pierre
7. Fourquet Jean-Yves
8. Contejean Evelyne
9. Bondon Pascal
10. Flaus Jean-Marie
11 ex. Andreoli Jean-Marc
11 ex. Jeron Thierry
11 ex. Queinnec Isabelle

*** Admission :
1. Ghorbel Fathi
2. Laboissiere Rafael
3. Desbat Laurent
4. Zasadzinski Michel

*** liste complementaire :
5. Peladeau Pierre
6. Queinnec Isabelle

707
Programmation, Genie Logiciel et Reseaux

Admissibilite :
1. Raymond Pascal
2. De Saqui Sannes Pierre
3. Contejean Evelyne

*** Admission :
1. Raymond Pascal

*** liste complementaire :
2. De Saqui Sannes Pierre

708
Informatique Industrielle

Admissibilite :
1. Cruette Didier

*** Admission :
pas d'admis, le candidat a retire sa candidature

709
Traitement du Signal

Admissibilite :
1. Bondon Pascal
2. Iouditsky Anatoli
3. Forster Philippe

*** Admission :
1. Bondon Pascal

*** pas de liste complementaire

710
Architecture

Admissibilite :
1. Giavitto Jean-Louis
2. Diot Christophe
3. Badouel Didier

*** Admission :
1. Giavitto Jean-Louis

*** liste complementaire :
2. Diot Christophe

711
Environnement programmation et parallelisme

Admissibilite :
1. De Groote Philippe
2. Ryan Mark
3. Domenjoud Eric

*** Admission :
1. Domenjoud Eric

*** pas de liste complementaire

712
Traitement d'images (Generation)

Admissibilite :
1. Bourdot Patrick

*** Admission :
1. Bourdot Patrick

713
Induction,decodage (Genome) (Imabio)

Poste non pourvu

714
Conception systemes techniques ouverts (PIRTTEM)

Admissibilite :
1. Lopez Pierre
2. Piechoviak Sylvain

*** Admission :
1. Lopez Pierre

*** pas de liste complementaire

715
Conception systemes techniques ouverts et formalisation comportements humains (PIRTTEM)

Poste non pourvu

716
Langage naturel, analyse et comprehension (Cognisciences)

Admissibilite :

1. Gaiffe Bertrand
2. Vieu Laure

*** Admission :
1. Gaiffe Bertrand

*** pas de liste complementaire

717
Bases de Donnees orientees objets / sequences biologiques (Imabio)

Admissibilite :
1. Perriere Guy

*** Admission :
1. Perriere Guy

Concours CR1:

703
Non Affiche
1 poste

Admissibilite :
1. Marzouki Meryem
2. Reed Bruce
3. Wang Dongming
4. Iouditski Anatoli

*** Admission :
1. Wang Dongming

*** liste complementaire :
2. Marzouki Meryem

704
Automatique des Procedes Biologiques

Admissibilite :
1. Flaus Jean-Marie
2. Queinnec Isabelle

*** Admission :
1. Flaus Jean-Marie

*** pas de liste complementaire

705
Modelisation du Raisonnement

Admissibilite :
1. Gregoire Eric
2. Bessiere Pierre
3. Herault Laurent

*** Admission :
1. Gregoire Eric

*** liste complementaire :
2. Bessiere Pierre

Compte rendu intersyndical de la session d'automne 1992 de la section 07 du comité national

6,7,8 Octobre 1992

M. Bayart (SNESUP), J. Bernussou (SNCS), C. Jard (SGEN)
D. Krob (SNCS), J.P. Laumond (SNCS), H. Prade (SNCS), X. Rousset (SNESUP)

Présents : M. Bayart, J. Bernussou, A. Costes, M.C. Gaudel, M. Jacobzone, C. Jard, P. Jorrand, J.P. Jouannaud, D. Krob (Secrétaire scientifique), J.L. Lacombe (Membre du bureau), J.P. Laumond, P. Lirou, O. Macchi, J. Mariani (Président), G. Mazaré, J.M. Pierrel (Membre du bureau), H. Prade, C. Puech, X. Rousset de Pina

Absents : P. Bernhard (excusé), M. Riquin (excusé)

1.1 Accueil du président

Le président excuse P. Bernhard qui sera absent ainsi que M. Riquin souffrant. Il présente ensuite l'ordre du jour et le calendrier prévisionnel de la session d'automne. Le procès verbal de la précédente session est ensuite adopté à l'unanimité.

Le président nous communique ensuite les dates des prochaines réunions de la section :

- 12 Février 93 : Bureau (session de printemps 1993)
- 24-25 Mars 93 : Session de printemps 1993
- 26 Mars 93 : Bureau (concours 1993)
- 10-11 Mai 1993 : Auditions (concours 1993)
- 12-14 Mai 1993 : Jury d'admissibilité 1993
- 13 Septembre 1993 : Bureau (session d'automne 1993)
- 5-8 Octobre 1993 : Session d'automne 1993

1.2 Exposé de politique générale du directeur de département

J.J Gagnepain, directeur du département, nous présente la politique du département SPI. Il commence par évoquer la rédéfinition en cours des OST. Il s'agit de redégager des priorités thématiques pour le département. Un comité de pilotage, constitué des présidents de section et de scientifiques extérieurs (J.C. Bermond, P. Bertrand), a été mis en place. Par ailleurs, une réunion de synthèse sur les OST réunissant tous les directeurs de laboratoires du département aura lieu à la Villette le 10 Décembre 1992.

J.J. Gagnepain évoque ensuite les projets européens. Il nous parle d'abord des projets "Capital humain et mobilité". Il nous fait part de son inquiétude sur les importantes modifications de classement (de l'ordre de 75% des dossiers) qui ont défavorisé tout particulièrement les demandes des laboratoires français pour ces projets. Par ailleurs, un grand nombre de dossiers "Réseaux" a aussi été déposé, mais il faut savoir que, compte tenu des contraintes budgétaires, le taux d'acceptation sera de l'ordre d'une demande sur 25 !

Le directeur du département nous rappelle également le nouvel organigramme du SPI. Celui-ci comprend maintenant 3 directeurs adjoints : B. Dubuisson pour les sections 7,8 et les relations avec la section 34, M. Champion pour les sections 9, 10 et les relations avec les section 4 et 22 et J.C. André pour les relations extérieures. Il y a aussi 6 chargés de missions, M. Dauchet s'occupant plus particulièrement de notre section.

J.J. Gagnepain nous fait aussi part du prochain déménagement des services centraux du CNRS. Ceux-ci sont actuellement dispersés sur 7 sites et devraient être regroupés rue Michel-Ange à partir de fin Juin.

Le directeur du département évoque ensuite la programmation des moyens. Il nous signale qu'une analyse de la distribution des postes de chercheurs montre que celle-ci a été faite sans réelle programmation. J.J. Gagnepain indique ensuite qu'il étudie si une certaine forme de contractualisation des laboratoires, permettant de garantir des moyens (avec leur évolution), voire des affichages, ne pourrait pas être envisagé. Une phase expérimentale pourrait être mise en place en 1993. Il nous signale par ailleurs qu'il convient de réfléchir sur le fait qu'un certain nombre de laboratoires de la section n'a ni ITA, ni chercheurs CNRS (15 laboratoires n'ont pas d'ITA CNRS et 12 n'ont aucun chercheur CNRS) J.J. Gagnepain nous fait aussi part du fait que la gestion des personnels ITA (hors ingénieurs de recherche) sera désormais du ressort des délégations régionales.

J.J. Gagnepain évoque ensuite la politique de l'emploi. Il nous signale que l'âge moyen du CNRS (45 ans) continue à croître. Le SPI quant à lui est un département plus jeune (moyenne d'âge : 40 ans) et le problème qui s'y pose est donc d'éviter son vieillissement. Pour cela, il compte essayer de voir dès 1993 quels seront les effets d'une politique de détachement sur ce problème.

J.J. Gagnepain nous signale aussi que plus de 75% du budget du CNRS a été consacré aux salaires en 1992. Il ne reste donc qu'une part de l'ordre de 25% des budgets qui peut être consacrée à une politique scientifique sur laquelle il faut en fait déduire un morceau très important pour les grands équipements et le soutien de base des unités. Le budget 1993 du CNRS devrait cependant être en progression de l'ordre de 4,5% par rapport à celui de 1992, mais seuls 2 postes d'ITA pourront être créé pour tout le CNRS !

Par ailleurs, le budget 1993 du SPI n'est pas encore fait. Le département peut encore se permettre de croître, mais dans les limites du budget disponible. Il convient donc d'examiner au niveau des créations d'unités, non seulement la qualité scientifique des laboratoires, mais aussi leur apport au dispositif général du CNRS (apport de nouvelles thématiques, insertion régionale, ...) J.J. Gagnepain évoque aussi le nombre de GDR du département (45 actuellement) qui est trop élevé. Il faudrait donc que le renouvellement d'un GDR au delà de 4 ans soit très exceptionnel, ce qui n'empêche pas d'arrêter un GDR pour en reproposer un autre dans la continuité.

J.J. Gagnepain évoque ensuite les deux nouvelles structures opérationnelles de recherche que sont les instituts fédératifs et les fédérations de laboratoires. La différence entre ces deux notions vient de ce que les instituts fédératifs ne concernent que les unités propres. Ces structures sont destinées à permettre de fédérer des unités CNRS, si possible sur un même lieu géographique. Les premières structures fédératives devraient être mise en place pour l'année prochaine. Dans notre section, cela devrait concerner l'IMAG pour lequel les liens avec le LIP sont encore à préciser. J.J. Gagnepain nous signale aussi qu'aucune position n'est encore arrêtée en ce qui concerne la participation de grosses équipes d'accueil de la DRED à des fédérations de laboratoires. Tout sera vu au cas par cas. A. Costes propose que le comité national émette un avis, le cas échéant, sur des demandes d'équipes DRED.

J.J. Gagnepain évoque alors un certain nombre de cas particuliers de laboratoires. Il nous demande d'examiner avec soin la situation des partenaires industriels en ce qui concerne les créations d'UMR. Il nous rappelle que la fin de la restructuration de TIM3 ne devrait pas poser de problèmes. Il nous signale également qu'il conviendra d'examiner l'IMAG tous les ans et qu'il faudra faire attention à ce que les laboratoires ne disparaissent pas derrière les projets de l'institut. Il évoque enfin la structure de l'IBP qui ne lui parait pas satisfaisante. M.C. Gaudel intervient aussi pour signaler qu'Imabio n'aurait pas apporté sa part au financement du GDR Génome et Informatique.

J.J. Gagnepain présente ensuite un rapide bilan des recrutements et des promotions pour l'année 1992 au niveau du SPI. Pour le passage CR2/CR1, tout le monde a été promu (32 promotions). Il y a aussi eu 16 promotions DR2/DR1 et 3 promotions DR1/DR0 au niveau du département. Enfin, il y a eu 51 recrutements en CR2, 14 en CR1 et 25 en DR2 pour le SPI.

En ce qui concerne les médailles, J.J. Gagnepain nous confirme donc l'obtention de la médaille d'argent par X.G. Viennot. Il nous rappelle qu'il n'est pas possible de proposer une équipe pour cette médaille. Il nous signale que le SPI devrait pouvoir aussi raisonnablement compter sur 2 médailles de cristal. Il évoque enfin le fait que la candidature de M.P. Schützenberger pour la médaille d'or n'est pas sortie au niveau le plus haut.

J.J. Gagnepain présente ensuite le bilan des détachements : J. Stern a donc été détaché sur un poste de DR1 et 3 personnes ont pu être détachées sur des postes de CR. Il nous rappelle qu'au niveau des promotions DR2/DR1, la règle de fonctionnement est la suivante : un certain nombre de postes est réparti suivant une base proportionnelle (2 pour notre section), le reste des postes étant attribué au mérite entre les différentes sections du département.

J.J. Gagnepain évoque enfin le concours 1993. Il semble qu'on puisse compter sur de l'ordre de 18 postes CR. 6 seraient affichés sur des thèmes propres (Réseaux hauts débits, Infographie, Automatique, Traitement du signal, Images et vision, Parallélisme) et 3 seraient des affichages doubles en lien avec Imabio, Cognisciences et le département SPM (pour un poste sur le thème Mathématiques et Informatique).

1.3 Bilan du concours d'entrée 1992

J. Mariani présente le bilan final du concours 1992. Pour notre section, 2 postes de CR2 et 1 poste de DR1 n'ont pas été pourvus. Il y a aussi eu une démission. Il évoque également les modifications de classement entre les jurys d'admissibilité et d'admission pour le concours 1992. Ceux-ci ont conduit en particulier au reclassement de D. Wang sur le poste CR1 et à la montée d'I. Queinnec sur la liste complémentaire des CR2. M. Marzouki qui avait été classé première sur le concours CR1 par le jury d'admissibilité a finalement été retenue sur la liste complémentaire de ce concours. J. Mariani rappelle également le fait que le jury d'admission n'a la possibilité de classer sur une liste complémentaire qu'au plus un nombre de personnes égal à la moitié des postes concernés par un concours.

J. Mariani évoque aussi les problèmes de passage de l'information entre les jurys d'admission et d'admissibilité ainsi que les problèmes de liaison avec l'INRIA. J.J. Gagnepain indique que le CNRS ne peut transmettre officiellement les résultats des jurys d'admission en raison d'un problème de confidentialité de ces résultats.

A. Costes signale que le poste DR1 non attribué a été utilisé pour permettre le détachement de J. Stern. Il demande les raisons de ce choix car ce poste aurait pu permettre un passage CR1/DR2 et un passage DR2/DR1 en plus. J.J. Gagnepain lui répond qu'il a considéré que le détachement de J. Stern était scientifiquement important. J.P. Laumond évoque les difficultés de carrière des jeunes chercheurs et qu'il conviendrait de mettre une priorité sur le déblocage CR1/DR2. M.C. Gaudel estime qu'il n'est pas clair à son avis que la priorité doive être mise sur les promotions et qu'il faudrait aussi penser en termes de passage vers l'enseignement supérieur, puis de retours temporaires éventuels au CNRS par le biais de détachements. J.J. Gagnepain indique aussi que les liens CNRS/Universités doivent être bidirectionnels. J. Mariani rappelle la situation particulière de notre section sur laquelle s'exerce la plus forte pression au recrutement par rapport au reste du CNRS.

1.4 Exposé du représentant de la DRED

J.P. Finance présente la situation de l'informatique à la DRED. Cette année s'achève la deuxième partie de la contractualisation de l'Ile de France et la fin de la contractualisation de l'Est et de l'Ouest. 2 nouveaux DEA ont été créés (soit 56 DEA au total pour notre secteur) et plus de 1500 thèses ont été soutenues dans le secteur SPI (sur 6000 thèses environ en France). L'informatique représente 5% des crédits au niveau de la DRED alors qu'elle forme près de 10% des thésards en France.

J.P. Finance rappelle que la DRED soutient près de 40 unités du CNRS, une cinquantaine d'équipes d'accueil et 6 jeunes équipes pour ce qui concerne notre section. Il nous fait aussi part de l'effet fédérateur très clair des écoles doctorales. Il évoque enfin les actions que la DRED soutient et en particulier les soutiens ponctuels qu'elle a donnée aux PRC par le biais d'actions incitatives. Le PRC Maths-Info a ainsi bénéficié d'un soutien supplémentaire de 550 kF grâce à la DRED. Il nous signale enfin que 45 postes d'ITA ont été créés cette année, dont 8 destinés au secteur SPI.

1.5 Formations en examen

La section examine ensuite les formations dans l'ordre suivant : UPR 175, UMR 14, UMR 17, GDR 39, GDR 973, GDR 1029, GDR 36, GDR 861, URA 227, URA 1095, URA 1439, URA 228, URA 317, URA 368, URA 369, URA 753, URA 749, URA 834, URA 820, URA 1118, URA 1376, URA 1377 et URA 1440.

Concernant les GDR 39 (Communication homme-machine), 973 (Architectures nouvelles de machine) et 861 (Circuits intégrés au silicium) qui sont tous à la fois en examen et couplés à des PRC, la section adopte une position de principe quant aux conditions préalables à un potentiel renouvellement, par le biais de la motion suivante adoptée à l'unanimité :

Comme pour l'ensemble des GDR thématiques, la section 07 attire l'attention sur le fait qu'à l'échéance de sa période d'existence actuelle, le GDR devra démontrer ses capacités d'innovation thématique et structurelle, et d'ouverture, sans se satisfaire d'une simple continuité avec ses activités passées.

Il est décidé de joindre le texte de cette motion aux phrases votées par la section pour les trois GDR précédemment mentionnés.

1.6 Formations en renouvellement

La section procède ensuite à l'examen des formations en renouvellement. Elle les examine dans l'ordre suivant : UPR 206, UPR 3251, UPR 8001, UMR 115, URA 262 et URA 817.

La section émet un avis positif pour le renouvellement de l'association des unités suivantes : UPR 206, UPR 3251, UPR 8001, URA 262 et URA 817. Elle émet un avis négatif pour le renouvellement de l'association de l'UMR 115 et propose que cette unité soit mise en restructuration pour une période d'un an.

1.7 Situation des GDR-PRC

Le président propose d'essayer de dégager la position de la section sur l'avenir des GDR thématiques en informatique couplés à des PRC, avant d'aborder l'examen des GDR en renouvellement. Il donne la parole à M.C. Gaudel qui avait été chargée par J. Baixeras d'élaborer un rapport sur les PRC. M.C. Gaudel rappelle que deux missions avaient été données aux PRC : d'abord celle d'animer une communauté et ensuite celle

de faire germer des projets ambitieux et finalisés. Globalement la première mission a bien été assurée et la deuxième moins bien. Elle nous signale aussi que l'idée que les PRC doivent contribuer à de la recherche de base est désormais bien admise. M.C. Gaudel évoque ensuite l'évolution des PRC : il est prévu de garder le découpage des PRC tel qu'il est, mais aussi de revoir leur composition et de développer des actions inter-PRC bien définies. Un directoire devrait aussi se mettre en place pour gérer les PRC. Elle nous signale enfin que suite à son rapport, les PRC C^3, CHM et ANM devraient être finalement financés sur la même base que les 4 autres pour l'année prochaine.

B. Dubuisson intervient alors en resoulignant le fait qu'il y a de gros problèmes concernant la survie des PRC et que les PRC doivent se transformer en opérations finalisés s'ils ne veulent pas être condamnés à disparaître. A. Costes propose alors de mettre en place au niveau du CNRS un comité inter-GDR permettant de coordiner l'action des différents GDR couplés à des PRC. Après une courte discussion, la motion suivante proposant la création d'un tel comité est votée par la section :

La section 07, après avoir analysé les GDR dans le domaine de l'informatique et leurs liens avec les PRC, propose la mise en place transitoire d'une commission inter-GDR avec la composition suivante :

- les sept directeurs de GDR-PRC
- H. Gallaire, M.C. Gaudel, M. Nivat, J.M. Pitie, G. Roucairol, M. Sintzoff

Résultat du vote sur la proposition : `14 Oui - 0 Non - 4 Abstention`

1.8 Examens des formations

La section procède ensuite à l'examen des GDR en renouvellement qu'elle examine dans l'ordre suivant : GDR 85, GDR 862, GDR 921 et GDR 922.

Compte tenu de l'âge avancé des GDR 85 (C^3) et 862 (Programmation) et de leur situation par rapport aux PRC, la section ne propose qu'un renouvellement exceptionnel de six mois pour ces structures de façon à leur permettre de déposer rapidement de nouveaux projets. La section émet également un avis défavorable en ce qui concerne le renouvellement des GDR 921 (Institut IMAG) et 922 (Institut Blaise Pascal), la structure de GDR ne lui semblant plus adaptée pour ces regroupements de laboratoires. Elle propose la création d'une fédération d'unités pour l'IMAG.

1.9 Prise en compte du tiers complémentaire

La section vote à l'unanimité la prise en compte du tiers complémentaire pour les chargés de recherche dont les noms suivent :

Nom	Prénom	Laboratoire
Bourdot	Patrick	UPR 3251 - Orsay
Boy de la Tour	Thierry	URA 1095 - Paris
Crepeau	Claude	URA 1327 - Paris
Giavitto	Jean-Louis	URA 410 - Orsay
Loeb	Elliot	URA 1304 - Bordeaux
Lopez	Pierre	UPR 8001 - Toulouse
Ortega Martinez	Roméo	URA 817 - Compiègne
Queinnec	Isabelle	UPR 8001 - Toulouse
Wang	Dongming	URA 394 - Grenoble

1.10 Nomination des directeurs de recherche des stagiaires

La section vote à l'unanimité les nominations suivantes des directeurs de recherche pour les chargés de recherche de 2ième classe stagiaires suivants :

Nom	Prénom	Laboratoire	Directeur de recherche
Bondon	Pascal	UMR 14 - Orsay	B. Picinbono
Bourdot	Patrick	UPR 3251 - Orsay	J. Mariani
Desbat	Laurent	ERS 6 - Grenoble	P. Cinquin
Demonjoud	Eric	URA 262 - Nancy	C. Kirchner
Gaiffe	Bertrand	URA 262 - Nancy	J.M. Pierrel
Giavitto	Jean-Louis	URA 410 - Orsay	M. Etiemble
Laboissière	Rafael	URA 368 - Grenoble	J.M. Dolmazon
Lopez	Pierre	UPR 8001 - Toulouse	J. Erschler
Péladeau	Pierre	URA 248 - Paris	D. Perrin
Perrière	Guy	URA 243 - Lyon	M. Gauthier
Queinnec	Isabelle	UPR 8001 - Toulouse	B. Dammon
Raymond	Pascal	URA 398 - Grenoble	J. Sifakis
Zasadzinski	Michel	URA 821 - Nancy	G. Krzakala

Elle vote également à l'unanimité les nominations suivantes de directeur de recherche pour les chargés de recherche de 1ère classe stagiaires suivants :

Nom	Prénom	Laboratoire	Directeur de recherche
Bessière	Pierre	URA 394 - Grenoble	P. Jorrand
Flaus Aubry	Jean-Marie	URA 228 - Grenoble	A. Cheruy
Gregoire	Eric	URA 262 - Nancy	J.P. Haton
Marzouki	Meryem	ERS 6 - Grenoble	B. Courtois
Wang	Dongming	URA 394 - Grenoble	R. Caffera

1.11 Demandes de subvention - Revues

Après classement des revues suivant leur audience et leur intérêt pour la communauté, la section vote à l'unanimité les propositions de subvention suivantes :

Titre	Subvention
Traitement du signal	40 kF
RAIRO Recherche Opérationnelle	25 kF
RAIRO Informatique Théorique	50 kF
Intellectica	10 kF
Mathématiques, Informatique et Sciences humaines	10 kF

1.12 Demandes de subvention - Colloques

Après classement des colloques en quatre catégories (internationaux, nationaux, groupes de travail, hors des thématiques de la section), la section vote à l'unanimité les propositions de subventions suivantes pour les colloques 1993 :

Titre	Lieu	Subvention
Tolérance aux fautes	Toulouse	30 kF
Optimisation des systèmes	Compiègne	15 kF
Journées Géométrie Algorithmique	Saint Pierre de Chartreuse	10 kF
EDAC/EUROASIC 93	La Défense	25 kF
Communication homme-ordinateur	Paris	10 kF
JAVA 93	Sophia-Antipolis	10 kF
TAPSOFT'93	Orsay	30 kF
Tableaux sémantiques et méthodes dérivées	Marseille	10 kF
Analyse numérique	Giens	0 kF
Courbes et surfaces	Chamonix	15 kF
Synthèse d'images	Paris	15 kF
Réseaux d'interconnexion	Lumigny	15 kF
EKAW'93	Toulouse	10 kF
IMDSP	Cannes	10 kF
IWPTS'93	Pau	15 kF
Société de Biomécanique	Paris	0 kF
Computing 93	Paris	15 kF
Robotique	Rodez	10 kF

M.C. Gaudel évoque rapidement le manque d'aide apporté par l'AFCET aux organisateurs de colloques internationaux en France.

1.13 Examen des demandes de création

La section examine les demandes de création d'unités dans l'ordre suivant : GDR Vaudry, USR Courtois, URA Demongeot, UMR Sifakis, URA Bourjault, URA Coquerez, URA Enjalbert, URA Florin, URA FRançon, URA Poinssac-Niel, URA Saucier, URA Praly, URA Laporte, URA Labeyrie, EP Puech et GDR Habib.

La section prend en considération toutes les demandes de création sauf le GDR Vaudry, le GDR Habib, l'URA Poinssac-Niel et l'URA Labeyrie. Elle refuse aussi de s'exprimer sur la prise en considération de l'URA Laporte.

P. Lirou évoque les problèmes de personnel qui pourraient apparaître en cas de mise en réaffectation de l'ERS 11 (correspondant à la demande d'URA Laporte). La section vote alors à l'unanimité la motion suivante :

Lorsque la restructuration ou la désassociation d'un laboratoire implique la modification de l'affectation des personnels chercheurs et ITA, la section souhaite qu'une cellule composée de M. Jacobzone et P. Jorrand puisse suivre les réaffectations de ces personnels.

Lors de la discussion sur l'URA Florin, M.C. Gaudel évoque les problèmes posés aux laboratoires parisiens par la création des nouvelles universités périphériques en Ile de France. Après un court débat, la section vote alors à l'unanimité la motion suivante à ce sujet :

La section 07 ne souhaite pas voir les laboratoires parisiens écartelés par la création des quatre nouvelles universités périphériques. Elle demande que les opérations liées à la création d'universités nouvelles soient menées en concertation avec le CNRS. Elle encourage les équipes de recherche de ces nouvelles universités à travailler avec leur nouvel environnement, de manière à préparer la création, à terme, d'unités associées viables.

1.14 Classement des unités en renouvellement et en création

La section vote à l'unanimité le classement suivant concernant les UPR en renouvellement :

1) (exaequo) UPR 206, UPR 3251, UPR 8001

Elle vote également à l'unanimité le classement suivant concernant les URA en renouvellement et en demande de création :

1) (exaequo) URA 817, URA 262, USR Courtois, URA Demongeot
5) URA Enjalbert
6) URA Praly
7) URA Françon
8) (exaequo) URA Bourjault, URA Coquerez, URA Florin, URA Saucier

Elle vote aussi à l'unanimité le classement suivant concernant les demandes de créations d'unités mixtes :

1) UMR Sifakis

Elle vote enfin à l'unanimité le classement suivant concernant les demandes de créations d'équipes postulantes :

1) EP Puech

1.15 Motion intersyndicale

D. Krob présente une motion au nom des élus syndiqués. Celle-ci évoque deux problèmes importants : elle demande dans un premier temps le rétablissement de la prime à 16% de l'indice moyen du corps pour les chercheurs; elle évoque ensuite le problème du blocage des carrières au CNRS avec notamment le verrou CR1-DR2. Une discussion s'engage alors sur la motion et tout particulièrement sur les solutions qu'elle proposait relativement au blocage des passages CR1-DR2. Il est finalement décidé de retirer du texte de la motion la partie relative à cette question. La section passe alors au vote et adopte à l'unanimité le texte suivant :

La situation du chercheur CNRS, du point de vue des salaires et des promotions, conduit à un sentiment de lassitude qui met en péril notre organisme et la cohésion des équipes les mieux soudées. En l'absence d'une véritable politique qui soit en accord avec les discours, il est urgent que la direction du CNRS prenne une mesure de sauvegarde. La proposition palliative suivante nous semble réaliste.

La prime des chercheurs instituée en 1959 est restée constante en francs courants pendant des décennies, jusqu'à sa récente revalorisation qui n'en modifie pas le caractère dérisoire (3 637 F annuellement pour un CR2). Depuis l'instauration de cette prime, la fonction du chercheur CNRS s'est considérablement diversifiée : gestion de contrats (qui conduit à l'amélioration à la fois du budget du CNRS et de son image), implication toujours grandissante des chercheurs CNRS dans les formations doctorales (fonction à juste titre reconnue pour nos collègues enseignants par une prime que les chercheurs CNRS ne perçoivent pas, alors même que leur ministre y est favorable), ... Le *rétablissement de la prime des chercheurs au taux antérieur de 16% de l'indice moyen du corps* constitue une mesure immédiatement applicable et pleinement justifiée par ces évolutions.

1.16 Rapport de conjoncture

J. Mariani rappelle que les membres de la section ont reçu les versions préliminaires (sauf en ce qui concerne le thème "Les mathématiques et leurs interactions") du rapport de conjoncture et que J. Bernussou est chargé de centraliser les remarques éventuelles de la section pour en faire part aux rédacteurs de la version finale du rapport. Il nous propose de nous réunir avant la fin novembre, date limite de remise des correctifs, avec P. Bertrand et J.C. Bermond qui ont aussi été chargés d'une réflexion sur le rapport de conjoncture, pour discuter de ce dossier.

1.17 Motion sur les conseils scientifiques

Le président nous signale que la direction du SPI prévoit pour des raisons budgétaires de n'organiser plus que des conseils scientifiques tous les 4 ans pour les unités de son

ressort. Après discussion, M. Bayart présente la motion suivante sur cette question qui est adoptée à l'unaninimité :

La section 07 souhaite qu'il y ait examen des laboratoires tous les deux ans avec en alternance un comité scientifique et un comité de suivi scientifique. Cette dernière réunion se ferait en présence de personnalités extérieures mandatées par le CNRS.

1.18 Fléchage d'un poste Maths-Info

Suite à la demande de la direction du département, la section examine la demande d'un fléchage d'un poste au concours d'entrée sur le thème "Maths-Info". Le président nous rappelle qu'il s'agit d'une suggestion qu'il a faite en concertation avec la présidente de la section 01 et le directeur du GDR-PRC "Maths-Info". La direction du département SPM serait d'accord pour donner sur cette thématique, un poste SPM évalué par notre section et destiné à un laboratoire SPM, pendant que le SPI ferait de même pour un poste qui serait destiné à un laboratoire de notre section, mais qui serait évalué en section 01.

La discussion s'engage ensuite. J.P. Jouannaud souligne les convergences scientifiques que l'on voit actuellement se dessiner entre mathématiques et informatique. M.C. Gaudel nous signale qu'à son avis, le terme "Maths-Info" ne recouvre pas une thématique scientifique. J.P. Jouannaud rappelle que les frontières entre mathématiques et informatique sur de nombreux thèmes (logique, complexité, ...) sont difficiles à cerner. Il propose de retenir les thèmes du PRC Maths-Info comme intitulé d'affichage. D. Krob signale que cette solution est aussi celle que souhaiterait le directeur du PRC, P. Flajolet. M.C. Gaudel propose que l'affichage soit plus ouvert et ait comme intitulé "Mathématiques et informatique". D. Krob suggère alors de retenir cet intitulé, mais suivi des thèmes du PRC. C'est finalement cette solution que retient la section qui vote alors la phrase suivante :

La section donne un avis favorable au fléchage d'un poste sans contraintes géographiques, sur la thématique suivante : "Mathématiques et Informatique (en particulier Algorithmique, Automates, Calcul formel, Combinatoire, Logique)".

Résultat du vote sur la phrase : `7 Oui - 0 Non - 3 Abstention`

1.19 Informations diverses

H. Prade signale qu'une réunion aura lieu à la mi-novembre à Paris pour évoquer la fusion du GIA/GRTC.

La direction du département nous signale qu'un nouveau décret va désormais permettre d'accélérer les promotions CR2/CR1. Celui-ci permet en effet de comptabiliser dans les 4 ans nécessaires à un CR2 pour être promouvable CR1, les années d'ancienneté qui ont été jugées équivalentes lors de sa reconstitution de carrière au moment du recrutement. Ce décret devrait être appliqué dès la prochaine session de printemps.

1.20 Jurys d'audition

Le président nous rappelle la façon dont les intitulés avaient été donnés aux différentes sections de jury d'audition l'an dernier. Il nous signale que les compositions de ces jurys seront faites lors du prochain bureau. Il suggère de prendre comme titre unique des 4 sections de jurys : "Sciences et technologies de l'information : informatique, automatique et traitement du signal".

1.21 Colloques et revues

C. Jard présente rapidement l'état des lieux qu'il a fait pour les revues et colloques concernant de notre section. Un décompte superficiel arrive déjà à plus de 300 revues. Ces listes seront diffusées à l'ensemble des membres de la section pour valider leur classement thématique et préparer leur regroupement en fonction de leur notoriété.

APPEL D'OFFRES DES PRC INFORMATIQUE

Appel d'offre des PRC Informatique

15 Octobre 1992

1 Introduction

Le Ministére de la Recherche et de l'Espace a créé et soutient depuis quelques années 7 Programmes de Recherches Coordonnées (PRC) en informatique; il s'agit des

- PRC Architectures Nouvelles de Machines (ANM)
- PRC Base de Données de 3ème Génération (BD3)
- PRC Communication Homme-Machine (CHM)
- PRC Communication, Concurrence et Coopération (C3)
- PRC Intelligence Artificielle (IA)
- PRC Mathématiques et Informatique (MI)
- PRC Programmation et Outils pour l'IA (PAOIA)

Ces PRC coordonnent leurs activités et mettent en place un appel d'offre commun pour 10 projets de recherche portant chacun sur l'un des 10 thèmes suivants:

1. Codage et cryptographie
2. Décision et apprentissage
3. Déduction automatique
4. Environnements de développement de Bases de Données
5. Gestion de l'incertain et de l'évolutif
6. Modèles fomels de calcul et langages
7. Spécification, vérification, et validation de programmes
8. Synthèse d'images et géométrie algorithmique
9. Systèmes à objets
10. Systèmes parallèles

Ces thèmes sont détaillés dans la suite de ce document..

Un comité inter-PRC, comprenant les directeurs des PRC et des personnalités extérieures, sélectionnera les projets retenus et gérera leur évaluation.

2 Modalités de soumission

L'appel d'offre est ouvert à toutes les équipes de recherche universitaires ou appartenant à des organismes publics de recherche (EPST ou EPIC) :

- l'appartenance aux PRC : n'est pas un préalable nécessaire pour soumettre un projet ou etre impliqué dans une soumission ;
- la taille des projets en nombre d'équipes et de chercheurs : typiquement 5 équipes faisant intervenir au total 25 chercheurs ;
- la durée des projets est de deux ans ;
- le soutien accordé à un projet est de 1MF TTC pour 2 ans;
- calendrier :

date limite de réception des soumissions 15 Novembre 1992,
publications des projets retenus : début Décembre 92

- forme des soumissions : il est important de bien clarifer les objectifs et de définir les approches qui seront poursuivis par le projet. Une soumission devrait typiquement aborder les points suivants :

(i) Objectifs scientifiques du projet : 1 page maximum.

(ii) Position des objectifs par rapport au thème affiché dans l'appel d'offres et à l'état de l'art (importance, caractère fortement innovant, problèmes ouverts) : 1 page maximum.

(iii) Approche(s) proposée(s), plan de travail pour atteindre les objectifs, à quel progrès s'attendre et comment l'évaluer : 4 à 5 pages maximum.

(iv) Intervenants dans le projet et leur role : 1 page maximum.

- adressez les soumissions en 3 exemplaires à l'un des directeurs des PRC

  - Daniel Etiemble, LRI, CNRS UA 410, Bat 490, Université de Paris Sud, 91405 Orsay cedex;
  - Philippe Flajolet, INRIA, Domaine de Voluceau, BP 105, 78153 Le Chesnay cedex;
  - Malik Ghallab, LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse cedex;
  - Jean-Paul Haton, CRIN, BP 239, 54506 Vandoeuvre les Nancy;
  - Patrice Quinton, IRISA, Campus de Beaulieu, 35042 Rennes cedex;
  - Patrick Sallé, IRIT , Université Paul Sabatier, 31062 Toulouse cedex;
  - Patrick Valduriez, INRIA, Domaine de Voluceau, BP 105, 78153 Le Chesnay cedex;

ou à : Marie-Claude Gaudel, LRI Bat 490, Université de Paris Sud, 91405 Orsay cedex

3 Description des Thèmes

3.1 Thème 1 : Codage et compression des données

L'objectif principal de cette opération est de fournir des percées algorithmiques dans le domaine de la transmission efficace et sure de très gros volumes de données. Ceci pose des problèmes de représentation et de compression qui sont à régler par des méthodes spécifiques au domaine des données, puis des problèmes de codage adaptés à la transmission et visant à minimiser les erreurs. Enfin la sécurité des informations transmises doit être assurée par des procédés cryptographiques spécifiques.

3.1.1 Codage et compression des données

L'utilisation de codages adaptés aux données à représenter doit permettre des gains dans des facteurs variant de 2 (texte) à 50 environ (données très corrélées). Une algorithmique spécifique sera développée à cette occasion portant notamment sur les points suivants:

- Données textuelles en langue naturelle.
- Représentation de contours d'images par méthodes formelles, notamment codages de contours, analyse syntaxique d'images, et représentations par automates.
- Codages d'adaptation aux contraintes de canaux. Il s'agit là d'élaborer des codes en longueur variable satisfaisant à des contraintes d'origine physique (magnétisation, erreurs de lecture) et évitant de longues monotonies de zéros et de uns dans les codages binaires.
- Codages de données numériques, par exemple séquences temporelles, en utilisant les redondances et les régularités sous-jacentes aux données.

3.1.2 Codes correcteurs d'erreurs

La construction de codes correcteurs d'erreurs à haute capacité de correction, par exemple pour des lignes très bruitées, est un domaine de haute technologie mathématique qui possède de nombreuses applications dans le domaine militaire et spatial. Les points à développer prioritairement sont les suivants:

- Codes issus de la géométrie algébrique. Il s'agit d'abord d'exploiter de manière concrète des bornes générales quant à la capacité de codage. Il s'agit ensuite d'étendre les codes correspondants pour couvrir tout le domaine du codage. Puis de trouver des codes spéciaux qui se prêtent à des algorithmes de décodage efficaces. Ceci nécessite de rendre constructif nombre de méthodes actuelles qui sont non-constructives.

- Codes analogiques. Il est nécessaire de concevoir des codes qui prennent en compte le caractère analogique des informations à l'origine digitales qui sont reçues, ce de sorte à maximiser les probabilités de décodage avec succès.

- Méthodes de constructions de codes correcteurs. Le problème est double: d'une part, il faut se doter de méthodes permettant de combiner diverses constructions de codes fondées sur un petit nombre de structures mathématiques (géométries finies, corps finis, empilements de sphères); d'autre part ces constructions complexes doivent conduire à des méthodes algorithmiques de décodage qui soient rapides tout en optimisant les probabilités de correction.

3.1.3 Cryptographie

Cet axe se situe dans un domaine, la cryptographie, qui depuis une quinzaine d'années, connaît un développement considérable grâce aux développements matériels et à l'apparition de nouvelles approches mathématiques. Il s'agit de construire les bases théoriques permettant d'envisager la réalisation effective de systèmes cryptographiques présentant des fonctionnalités entièrement nouvelles: systèmes à clefs publiques et systèmes "zero–knowledge", dans le contexte de l'échange d'informations et de l'authentification sur un canal public. Les thèmes principaux abordés seront les suivants:

- Cryptanalyse des systèmes à clefs publiques. Problèmes algorithmiques fins de réduction des réseaux. Études de parallélisation des algorithmes correspondants.

- Réalisation d'algorithmes "zero–knowledge". Études d'implantation sur carte à mémoire.

- Robustesse des systèmes cryptographiques fondés sur des fonctions difficilement inversibles.

De telles études doivent s'appuyer sur les bases théoriques fournies par des développements récents de la théorie de la complexité (approximation de problèmes *NP*–complets, amplification d'erreurs dans les preuves) et sur l'algorithmique finement optimisée des structures algébriques fondamentales (nombres, corps finis, réseaux).

3.2 Thème 2 : Décision et Apprentissage

Les systèmes de saisie automatique, répandus dans de nombreux secteurs industriels, fournissent de larges masses de données de toutes sortes, aujourd'hui difficilement exploitables. Certaines de ces données sont essentiellement quantitatives et comportent quelques marqueurs symboliques ; d'autres sont principalement symboliques et font usage de quelques descripteurs numériques. Pour

des taches telles que la surveillance, le diagnostic ou l'aide à la décision, et sur des applications allant du controle de processus industriels ou du controle aérien à la classification en biologie moléculaire ou au séquencage du génome humain, il est essentiel de pouvoir extraire de ces données symboliques et numériques des informations utiles, interprétatives ou explicatives. Cette extraction exige presque systématiquement une étape préalable d'analyse des données, d'acquisition de modèles et de connaissances, d'apprentissage.

Il semble clair aujourd'hui que pour ce type de problèmes, l'interprétation et la décision tout autant que l'apprentissage ne peuvent s'appuyer sur des approches exclusivement numériques ou purement symboliques. Il est essentiel de pouvoir combiner étroitement dans le raisonnement des traitements symboliques et des traitements numériques.

Le souci d'union des approches symboliques et numériques est une tendance dont on peut observer le développement dans la communauté internationale comme en témoigne l'existence de conférences ou d'ouvrages spécialisés. En France, plusieurs conférences sur le thème "Apprentissage Symbolique-Numérique" ont eu lieu ces dernières années.

Le projet de recherche qui sera soutenu sur ce thème focalisera l'activité de plusieurs équipes sur une approche ambitieuse des problèmes d'apprentissage, d'interprétation ou de décision, combinant Le projet de recherche qui sera soutenu sur ce thème focalisera l'activité de plusieurs équipes sur une approche ambitieuse des problèmes d'apprentissage, d'interprétation ou de décision, combinant des méthodes numériques et symboliques. Il s'agira en particulier d'approfondir un ou plusieurs des points suivants :

- méthodes connexionnistes mixtes numériques-symboliques : par exemple, exploration des techniques symboliques pour la définition de structures de réseaux neuronaux "intelligibles", ou des techniques neuronales pour l'acquisition d'une partie de l'expertise (paramètres, coefficients, seuils), extraction des règles équivalentes à un réseau pour des besoins d'explication ou d'argumentation;
- méthodes d'analyse des données, étendues à la prise en compte d'informations symboliques (e.g., attributs des classes et leurs liens par les axiomes du domaine), controle de l'analyse des données (choix de modèles et méthodes) par l'expertise générale ou spécifique, méthodes de type "classification structurelle" pour la structuration ou la généralisation;
- méthodes de raisonnement symboliques s'appuyant sur l'information numérique à différents niveaux: qualifications numériques de faits ou de règles (e.g. inférence floue), association de distributions à des relations symboliques (e.g. réseaux probabilistes), ou utilisation de distances numériques pour guider le raisonnement (heuristiques) ou mesurer des proximités ou des analogies;
- validation, qualification de méthodes symboliques, en particulier d'apprentissage ou de classification, par des techniques numériques, e.g., validation statistique de l'apprentissage, ou analyse de convergence, modélisation probabiliste, analyse de la robustesse ou de la stabilité d'un réseau probabiliste ou d'un système de règles.

Le projet soutenu s'attachera à caractériser les méthodes développées et les approches analysées relativement à des classes d'applications efficacement résolues .

3.3 Thème 3 : Déduction automatique

La logique, classique et non classique, est très présente en informatique, et en IA plus particulièrement. Elle se manifeste aussi bien au niveau des fondements théoriques de ces disciplines qu'au niveau des représentations et des techniques utilisées pour la formalisation et la résolution d'une large classe de problèmes. Par ailleurs, ces disciplines ont conduit à un renouveau de la logique mathématique et à un enrichissement de ses problématiques.

L'intéret pour la mécanisation de la logique est multiple et occupe une communauté importante sur le plan international. En Intelligence Artificielle, il s'agit de réaliser des systèmes d'inférence, et d'en établir les propriétés syntaxiques, sémantiques et algorithmiques et de caractériser les classes de problèmes qui peuvent etre résolus par de tels systèmes. Il s'agit également de modéliser des raisonnements complexes, mettant en oeuvre par exemple des modalités et des valuations, par des théories logiques diverses et d'établir des liens entre ces théories. Il s'agit enfin de développer des systèmes de programmation de type logique puissants et performants.

La déduction automatique trouve également un domaine d'application privilégié dans les bases de données déductives, ou' des règles de déduction (en général pas trop nombreuses) cohabitent avec des faits (eux, nombreux). Selon le type de règles (règles générales, contraintes d'intégrité, règles exprimant les droits d'accès, et de modification pour la sécurité des données,...) les logiques sous-jacentes sont diverses (logique du premier ordre, logique modale, logique temporelle, logique déontique,...) et les problèmes de déduction automatique plus ou moins complexes.

Mentionnons enfin les applications de la déduction automatique à la vérification des systèmes informatiques (logiciel, matériel et réseau de communication), ainsi qu'au calcul formel.

Le projet de recherche qui sera soutenu sur ce thème focalisera l'activité de plusieurs équipes sur une approche ambitieuse de la déduction automatique, dans les logiques standards (classique, intuitionniste, linéare) et non standard (modales, multivaluées), avec des objectifs de comparaison et d'intégration, éventuellement d'unification. Il s'agira en particulier d'approfondir un ou plusieurs des points suivants :

- liens entre les techniques de déduction dans les logiques classiques et non classiques (e.g. modale), possibilités de traduction des secondes vers certaines théories des première et étude de leurs propriétés en termes algorithmiques et de représentation des connaissances ;
- liens et complémentarité entre diverses approches à la déduction automatique : syntaxiques (e.g. résolution, réécriture, calcul des séquents) et sémantiques

(e.g. tableaux), cas des logiques multivaluées, ou non-monotones ;

- intégration de diverses approches à l'un des système ou "ateliers" de déduction automatique qui sont développés en France, en particuliers ceux "ouverts" (paramétrables, multi-logiques), regroupement des efforts en la matière ;

- interaction utilisateur-système de déduction automatique : représentation de preuves, simplification, transformation, analogie entre preuves, construction de contre-exemples, explication, analyse de l'échec, traduction et exploitation de l'expertise du domaine ;

- caractérisation, autour d'un domaine d'application, de classes de problèmes efficacement résolus, en terme de complexité théorique et pratique.

3.4 Thème 4 : Environnements de Développement de Bases de Données

L'accroissement constant des performances des matériels permet le développement de logiciels très sophistiqués. Ceux-ci peuvent faire intervenir différents composants complexes tels que compilateurs, déboggeurs, bases de données, etc., qu'il est nécessaire de voir intégrés au sein d'un même environnement de développement d'applications. Les objectifs visés par de tels environnements sont l'augmentation de la productivité des développeurs, la fiabilité afin de garantir la sécurité des applications et les performances des applications générées. La productivité peut être augmentée en utilisant des langages de haut niveau (déclaratifs) reposant sur un paradigme formel (logique, fonctionnel, etc.) permettant l'accès transparent aux données persistantes. La fiabilité peut être assurée en s'appuyant sur un système de typage fort qu'il convient de ne pas limiter en puissance. Enfin, les bonnes performances des applications générées reposent sur l'optimisation et l'utilisation de structures de données efficaces et, de facon complémentaire, la parallélisation du code pour exécution sur multiprocesseur et le support par le système d'exécution.

Les environnements de développement visés par ce thème de l' appel d'offres sont des logiciels expérimentaux et démontrables. Les travaux concernent l'utilisation des paradigmes fondamentaux mis en oeuvre dans la conception d'outils avancés afin de réaliser des environnements performants pour des domaines d'application spécifiques.

Les paradigmes fondamentaux qui seront étudiés en vue de leur utilisation pratique sont:

(1) Les spécifications formelles comme un mécanisme de base pour spécifier et expliquer le comportement des environnements (spécifications algébriques, en logique, par contraintes, etc.).

(2) La sémantique et les systèmes de preuves pour définir, exécuter et prouver des propriétés de programmes (sémantique déclarative et opérationnelle, théorie des types, etc.).

(3) L'approche à objets pour unifier des composants hétérogènes et représenter des données à structure complexe ou évolutive.

(4) Les architecture logicielles modulaires et réparties (client/serveur, mémoire virtuelle distribuée, etc.).

Les environnements spécialisés et les outils associés qui seront développés sont:

(1) Les compilateurs, optimiseurs, paralléliseurs et interpréteurs pour langages intégrant l'accès aux bases de données déductives et objets.

(2) Les outils d'aide à la conception logique et physique des bases de données déductives et objets tenant compte de l'évolutivité des applications.

(3) Les outils d'aide au développement d'applications exploitant l'interactivité, la réutilisation, l'incrémentalité et la navigation dans des objets comme principes de base de communication homme-machine (hypertexte, hypermédia, etc.).

3.5 Thème 5 : Gestion de l'incertain et de l'évolutif

Dans de nombreuses classes d'applications, on ne dispose que d'informations incomplètes, incertaines, voire meme partiellement incohérentes. C'est le cas par exemple pour un système de surveillance ou de controle de processus, à partir d'une instrumentation partielle, bruitée et pouvant avoir certains capteurs défaillants. C'est le cas également pour un gestionnaire de base de données issu de la fusion de plusieurs sources. Il est nécessaire de tirer des conclusions de telles informations, d'en qualifier la vraisemblance, et de tenir compte de leur caractère provisoire, et de leur éventuelle remise en cause par l'arrivée d'informations complémentaires, ou par l'évolution dans le temps du système modélisé.

La formalisation de raisonnements sur des informations incertaines et évolutives a fait l'objet de nombreux travaux en Intelligence Artificielle. Les approches proposées sont en apparence très diverses, certaines sont purement symboliques et traitent qualitativement l'incertitude, d'autres utilisent des nombres pour la quantifier. Certaines cherchent à formaliser le raisonnement plausible et à traiter les exceptions. Elles ont donné lieu à de nombreuses études sur la non monotonie. D'autres ont abordé directement les problèmes de révision de bases de connaissances à l'arrivée de nouvelles informations contredisant une partie du contenu de ces bases.

Avec l'extension de l'emploi des outils forgés pour le raisonnement non monotone, la notion d'information incomplète s'élargit, et non monotonie et révision apparaissent comme deux approches de problèmes semblables. On observe ces dernières années, un regain d'intéret pour l'approche révision, du probablement aux difficultés des techniques non monotones. En effet, bien que nombreuses et variées les voies offertes par l'approche "non monotone" n'ont pas réellement débouché. L'approche "révision" sur laquelle ce thème se focalise, apparait plus souple et plus puissante en meme temps ; on peut espérer qu'elle apportera davantage de réponses à ces problèmes. D'autre part, elle s'attaque à une problématique plus large, qui est la représentation de situations évoluant dans le temps. Ce dernier aspect est important pour les I ıses de Données, et recouvre le problème des mises à jour d'une base, ainsi qu'en Intelligence

Artificielle, dans le cadre de la théorie des actions et du raisonnement sur le changement.

On peut considérer qu'un processus de révision d'une base de connaissances consiste à modifier cette base pour prendre en compte l'arrivée de nouvelles informations. Un tel processus peut etre envisagé dans deux types de situations différentes conduisant ainsi à deux problématiques quelque peu distinctes : l'évaluation des croyances d'un agent (ou d'un système informatique) d'une part, et la description des changements intervenant dans un monde évolutif, d'autre part.

Dans le premier cas, le système maintient des croyances sur la vérité ou la fausseté de propositions sur lesquelles l'information manque au moins partiellement ou est empreinte d'incertitude. Pour tirer des conclusions de telles propositions, on complète en général l'information manquante grace à la connaissances de ce qui se passe dans des situations normales ou typiques. On fait l'hypothèse qu'une situation exceptionnelle est accompagnée de l'information qui permet de détecter l'exception. L'arrivée de nouvelles informations, si elles vont à l'encontre de cette hypothèse, oblige à une révision des croyances courantes maintenues par le système. Dans ce cas l'idée d'ordre sur les normes (éventuellement exprimé numériquement) apparait naturellement, certaines choses pouvant etre davantage crédibles que d'autres. Cet ordre permet de guider la procédure de restauration de la cohérence nécessitée par l'arrivée de nouvelles informations, en éliminant les énoncés les moins crédibles qui créent cette incohérence. Ainsi, le processus de la révision cherchera à maintenir ou à augmenter la qualité des arguments qui pourraient etre fournis à l'appui des conclusions du système. Sur le plan technique, le diagnostic de pannes en environnement non-évolutif, ou' les informations sur le système défaillant ne sont pas disponibles au meme moment ou au meme cout, offrent un bon exemple de ce type de processus. D'un point de vue général, la gestion de la normalité renvoie à la problématique de la typicalité, bine connue en Sciences Cognitives.

Dans la seconde situation, la description du monde, sur lequel on dispose en général d'une information certaine et relativement complète, doit etre mise à jour à l'arrivée d'informations vraies sur le nouvel état du monde, mais contradictoires avec la description que l'on en avait. Ces nouvelles informations peuvent décrire les conséquences d'une action effectuée ou refléter l'observation d'un changement. Elles peuvent également traduire l'apparition d'une nouvelle contrainte à prendre en compte dans la résolution d'un problème. Ce qui importe alors est d'etre fidèle à l'évolution. L'aspect chronologique du problème est clairement important dans ce suivi de l'évolution du monde. Par ailleurs, la nécessité d'un retour à une consistance forte dans le passage d'un état à un autre est manifeste dans cette problématique puisque la description du monde doit rester cohérente, alors qu'il est éventuellement plus acceptable d'avoir des incohérences partielles ou "locales" dans un état de croyance.

Cette distinction entre les deux types de situation n'est pas toujours évidente. Par exemple, le diagnostic en temps réel de processus évolutifs, relève des deux

problématiques à la fois. Il en va de meme du raisonnement sur le chargement à partir d'informations partielles ou incertaines. La différence réside essentiellement dans le caractère plus ou moins complet, plus ou moins certain des informations, et dans le caractère statique ou dynamique du monde à modéliser.

Le projet qui sera retenu sur ce thème portera sur l'un ou les deux volets de la problématique de la révision. Il s'attachera à développer de nouvelles méthodes ou à étendre les méthodes existantes, aussi bien syntaxiques que sémantiques, pour la révision. Ces méthodes seront caractérisées au niveau de leurs propriétés et mises en oeuvre informatique. Elles seront comparées entre elles et confrontées à des classes d'application, en IA et en Bases de Données.

3.6 Thème 6 : Modèles de calcul et Langages

La programmation s'est très diversifiée mais elle reste la motivation centrale de l'informaticien. Ce thème concerne les fondements théoriques et l'étude des langages de programmation et des modèles qui leur sont associés. Abstraction, modularité, puissance d'expression, structuration, efficacité, réutilisabilité, possibilité de manipulation formelle et indépendance par rapport aux supports matériels d'exécution sont parmi les objectifs des concepteurs de langages. La sémantique, les techniques d'interprétation et de compilation, les théories de types associées sont les outils pour atteindre ces buts. Les sous-thèmes que nous proposons sont les suivants:

- programmation logico-fonctionnelle

  Le domaine évolue dans deux directions de recherche assez distinctes. La première est l'intégration des styles de programmation fonctionnelle, équationnelle et logique, en particulier par l'intégration de termes fonctionnels dans les clauses logiques. La deuxième est l'intégration de contraintes en programmation fonctionnelle et logique qui peuvent prendre trois formes:

  - contraintes booléennes et "boolean decision diagram" comme outil efficace d'évaluation,
  - contraintes numériques, domaine lié à la géométrie algorithmique et non à la logique,
  - contraintes symboliques portant sur des termes d'algèbres libres quotient.

  Ce tout dernier point a vu récemment des progrès rapides et des techniques nouvelles et prometteuses qui intègrent logique et systèmes de réécriture. Ce sont des techniques unificatrices et simplifiantes: interprétation des sortes comme ensembles réguliers de termes pour prendre un bon exemple.

- théories de types

  Calculer le type d'un programme est une preuve de correction partielle et un type peut être également vu comme une spécification. Des études sur les diverses formes de polymorphisme sont en cours. Elles font appel aux catégories et la logique linéaire notamment pour résoudre le problème de l'affectation dans les langages fonctionnels impurs.

  La théorie des types dont le point de départ est l'isomorphisme de Curry Howard entre lambda-termes et preuves intuitionnistes est maintenant considérée comme un outil conceptuel incontournable pour la réflexion sur la programmation (voir les récents colloques "Principles of Programming Languages"). Elle concerne la définition des langages de programmation, les preuves de validité, la méthodologie de programmation, la synthèse de programmes à partir de preuves (théorie de la réalisabilité), les spécifications de programmes. Il est apparu récemment que cet isomorphisme pouvait s'étendre aux preuves en logique classique et aux programmes impératifs avec affectations et sauts. C'est une direction nouvelle dont les conséquences sont à explorer. La logique linéaire est un autre exemple d'une synergie exemplaire entre théorie de la démonstration et informatique. Ce formalisme, qui résulte d'une analyse des preuves en logique classique à la lumière de la notion informatique de ressource limitée est un cadre conceptuel important et prometteur pour la sémantique des langages de programmation. Cette logique a beaucoup d'autres points d'intérêt: intégration d'une dynamique dans les systèmes de preuves, obtention de *la structure optimale* pour l'évaluation des lambda-termes (un des résultats les plus forts obtenus récemment), inspiration de la déduction libre et dévelopements de la théorie de la démonstration.

- fondements théoriques

  Etudes des modèles sous-jacents aux outils de spécification et de vérification, tels les calculs de processus, ordres partiels, structures d'évènements, systèmes de transitions, traces d'exécutions.

- programmation par objets

  La programmation par objets commence à pénétrer dans les milieux industriels alors que des problèmes théoriques et pratiques ne sont pas encore totalement éclaircis. Il faut donc poursuivre les études théoriques sur le modèle objet, sur les mécanismes d'héritage, la réflexivité, le typage statique et dynamique des méthodes. L'application au développements de grands projets nécessite la définition de méthodes de programmation et de vérification et l'étude du rapport entre héritage et réutilisation. Enfin le problème de la persistance doit être étudiéen collaboration avec les chercheurs en bases de données.

- programmation parallèle

Cet axe concerne d'une part la parallélisation automatique des programmes séquentiels. Dans le cas des langages fonctionnels et logiques les sources du parallélisme sont nombreuses et il faut savoir en extraire le bon parallélisme. Ceci peut se faire par exemple par une analyse de la stricticité et des transformations de programme. Un autre axe concerne la définition et l'implantation efficace de modèles de calcul parallèle: acteurs, réduction de graphes, machine chimique, ...

- langages synchrones

  Les langages synchrones permettent la programmation des systèmes réactifs: contrôles de processus temps réél, automatismes, contrôleurs matériels et logiciels, protocoles de communication, traitement du signal, etc. Ces langages reposent sur une hypothèse de synchronisme parfait et permettent d'écrire des programmes à la fois parallèles et détreministes.

- compilation et interprétation

  Comme le thème précédent celui-ci est fortement lié aux GDR/PRC C3 et ANM. Il concerne toutes les techniques de compilation et d'optimisation de code: métacompilation et sémantique formelle, rapports entre compilation, interprétation et évaluation partielle, interprétation et compilation dans un contexte réparti, problémes de gestion mémoire notamment ramassages de miettes dans les langages à objets et/ou répartis.

3.7 Thème 7 : Spécification Vérification Validation

La maîtrise de la production de logiciels ne peut s'envisager qu'au travers d'une nécessaire formalisation de cette production. Cette réalité, affirmée depuis longtemps par les chercheurs, est aujourd'hui de plus en plus largement reconnue par les industriels. Ainsi, même dans le domaine souvent empirique de la production de logiciels, il est clairement indispensable de poursuivre des recherches fondamentales sur différents formalismes. Ces recherches permettront de mieux maîtriser la production de logiciels complexes, et ceci durant toutes les étapes du cycle de vie du logiciel. Dans l'activité de développement de logiciels, qu'il comportent ou non du parallélisme, les étapes initiales de spécification et de conception sont maintenant de plus en plus reconnues comme essentielles. Une des raisons de cette importance est leur impact sur les activités de validation et de vérification et sur la qualité (au sens grand public du terme) des logiciels obtenus. Dans ce domaine les recherches sur les approches formelles, très actives depuis des années, ouvrent maintenant des perspectives très prometteuses. Ces perspectives sont fondées sur des avancées constatées dans différents domaines :

- Des résultats importants ont été obtenus sur la sémantique du parallélisme et sur la sémantique des types de données, ainsi que sur la notion d'observabilité qui s'est avérée être un concept clé pour tout ce qui concerne le raffinement.

- Divers modèles sémantiques existent, aussi bien pour le parallélisme que pour les types de données. Dans le cadre du parallélisme, moins le modèle est structuré, plus les outils de validation et de vérification sont opérants ; mais ils ne permettent de traiter que des modèles de petites tailles. Des avancées décisives fondées sur la modularité ont par contre été obtenues dans le domaine des types de données et devraient pouvoir rejaillir, entre autres, sur la conception d'outils d'analyse statique de systèmes parallèles.

- Les systèmes informatiques sont suffisamment diversifiés et complexes pour qu'il paraisse naturel d'envisager plusieurs formalismes pour spécifier, valider et vérifier un même système, selon le type de propriétés auxquelles on s'interesse. Les recherches sur ces différents formalismes sont maintenant mûres pour être confrontées et pour approfondir des recherches sur leurs complémentarité et leur usage coordonné.

- Dans le domaine des langages de spécification, la situation est en train de se stabiliser autour d'un certain nombre de grandes approches qui sont reconnues comme complémentaires, même si cette complémentarité reste un sujet de recherche.

- Dans le domaine de la validation et de la vérification, des progrès significatifs ont été accomplis en ce qui concerne les théories de preuve et de test dans les différentes logiques qui interviennent dans le développement de logiciel ; simultanément, la croissance spectaculaire des puissances de calcul a permis le développement et l'utilisation d'outils tout à fait crédibles.

- L'utilisation de la correspondance entre preuves et programmes permet la construction automatique par la machine d'un programme à partir de sa spécification. Cette analogie demande encore des recherches afin d'améliorer les programmes produits et de construire les environnements de développement correspondants.

3.8 Thème 8 : Synthèse d'images et géométrie algorithmique

L'objectif principal de cette opération est le développement de méthodes de nature géométrique et algorithmique permettant le traitement de structures géométriques complexes. C'est ensuite l'utilisation des méthodes élaborées à fin de visualisation de ces structures dans des conditions de très haute performance. Il est notamment prévu de prendre en compte les aspects tant statiques que dynamiques des objets manipulés et un certain nombre d'applications seront traitées dans le domaine du rendu réaliste de scènes tridimensionnelles.

3.8.1 Algorithmique géométrique

Il s'agit d'abord de développer des méthodes efficaces de traitement de données géométriques d'utilisation générale: triangulations, diagrammes de Voronoi, maillages avec contraintes. Une algorithmique probabiliste fondée sur l'échantillonage aléatoire ("randomisation") et des structures de données adaptées doivent être conçues pour assurer ces fonctionnalités.

Les aspects incrémentaux (mise à jour des représentations) nécessitent de prendre en compte la cohérence spatiale et temporelle des structures manipulées, ce qui pose des problèmes de conception algorithmique d'un type tout à fait particulier. Le critère d'efficacité essentiel, non classique, doit être largement fondé sur l'amortissement du coût des opérations sur un ensemble complet de traitements.

Les problèmes de visibilité imposent de développer une algorithmique capable de traiter des ensembles géométriques complexes: grand nombre d'objets, surfaces algebriques, espaces de dimension élevée.

3.8.2 Algorithmique graphique

Une première activité consistera à bâtir un corpus de primitives de base (donc de portées générales et réutilisables ainsi qu'éventuellement susceptibles d'implantations materielles nouvelles) pour la manipulation et le rendu des structures fondamentales (lignes, courbes, arbres, graphes, cartes) au moyen notamment des méthodes de la géométrie discrète.

Les applications visés sont dans le domaine de l'imagerie de haute qualité, dans la synthèse d'images réalistes, dans le domaine de la visualisation de dessins ou schemas complexes.

Par ailleurs, on développera des systèmes permettant de modéliser des objets tout en disposant d'un contrôle suffisamment riche de leur topologie ou de leur géométrie.

3.8.3 Synthèse d'images

Les algorithmiques géométrique et graphique constituent les bases sur lesquelles construire les représentations internes de structures complexes.

La simulation de systèmes mettant en jeu ces éléments, demande l'élaboration et la mise en oeuvre de modèles dont les plus prometteurs reposent aujourd'hui sur des modèles dits "physiques" (domaine de la simulation de l'éclairage ou de l'animation, par exemple). On s'attachera à développer des modèles, des algorithmes, des systèmes permettant de construire, puis de simuler, en des temps "raisonnables" des objets, des scènes, des phénomènes, des mouvements, d'une complexité significative. Les applications médicales peuvent, en particulier, fournir des applications intéressantes de ce point de vue.

3.9 Thème 9 : Systèmes à Objets

Le développement de nouvelles applications interactives complexes (CAO, bureautique, géomatique, etc.) nécessite la conception et la réalisation de systèmes à objets complexes, partagés et persistants. Ces systémes mettent en jeu des méthodes et des techniques pour stocker les objets, les retrouver efficacement, les rendre persistants et résistants aux pannes et permettre leurs accès et leurs mises-à-jour concurrentes par de multiples utilisateurs. D'une facon générale, l'étude des systémes à objets doit impérativement prendre en compte le problème de la cohérence des objets en tenant compte de leurs spécificités (objets typés, relations, etc.), dans le contexte d'environnements répartis ou parallèles, sujets aux pannes.

Les systèmes visés par cet appel d'offres sont des logiciels expérimentaux et démontrables. Les travaux concernent l'étude et l'utilisation des paradigmes fondamentaux de conception de systèmes à objets répartis et leur application dans la construction et l'évaluation de prototypes.

Les paradigmes fondamentaux qui seront étudiés sont:

(1) L'extension de la théorie de la sérialisabilité et son application aux mécanismes de gestion de concurrence d'accès aux objets répartis (transactions entrelacées, atomicité des opérations, versions, etc.).

(2) La résistance aux pannes dans les systèmes distribués ou parallèles (contraintes temps réel, utilisation de mémoire stable, journaux, etc.).

(3) La modélisation et la simulation des systèmes à objets distribués ou parallèles pour determiner leur comportement en termes de performances et de fiabilité de fonctionnement.

(4) Les structures de données et algorithmes associés pour modéliser et supporter efficacement les objets complexes (objets spatiaux, placement de graphes d'objets répartis, glanage réparti, etc.).

(5) Le support apporté par le système à la gestion d'objets persistants et répartis : mémoire persistante répartie, mécanismes d'accès et de localisation, etc, compte tenu notamment de l'évolution technique prévisible (grands espaces d'adressage, réseaux rapides) ; les méthodes de gestion de la répartition (objets fragmentés, etc.).

(6) La prise en compte d'aspects liés aux langages de programmation pour la gestion d'objets (utilisation des informations de type, extensions persistantes de langages, etc.) ; les me'thodes à base d'encapsulation pour la réutilisation d'applications existantes et l'interopérabilité entre systèmes à objets.

Les systèmes à objets spécifiques qui seront développés sont:

(1) Les systèmes répartis à objets génériques (multi-applications) exploitant l'approche micro-noyau (type Chorus ou Mach), en vue de séparer les mécanismes de base des politiques, extérieures au noyau et adaptables 'a chaque cas.

(2) Les gestionnaires d'objets complexes de grande taille pour systèmes multi-média ou géographiques répartis ou parallèles.

(3) Les systèmes à objets répartis tolérants aux fautes, notamment pour des applications à fortes contraintes temps réel.

3.10 Thème 10 : Systèmes parallèles: Langages, Modèles d'exécution et Architectures pour le parallélisme massif

L'importance des machines massivement parallèles est maintenant incontestable pour répondre aux besoins croissants en puissance de calcul, en manipulation de très grosses bases de données et de connaissances, et en puissance de communication. Le parallélisme massif implique des machines à mémoires distribuées. L'expérimentation et l'utilisation de ces machines a commencé il y a quelques années. Différentes approches ont été utilisées : l'approche SIMD avec la CM-2 de TMC, l'approche MIMD à mémoire distribuée avec les hypercubes (IPSC2) d'Intel et les machines à base de Transputers. Une nouvelle génération de machines, s'appuyant sur la technologie RISC, apparait, comme la CM-5 de TMC, la machine Paragon d'Intel, ou en France les machines de la société Archipel. Ces machines utilisent l'approche "parallélisme de données", en mode SPMD. Certaines, comme la CM-5, combinent le parallélisme de données avec une approche vectorielle au niveau des processeurs élémentaires.

Les recherches menées depuis plusieurs années, en France et au niveau international, ont permis de clarifier les problèmes de la mise en oeuvre du parallélisme massif. On peut en énumérer le plus grand nombre de la manière suivante, selon les types d'architectures considérées, avec les points-clé pour chaque cas:

- architectures parallèles spécialisées:

  lien entre les architectures cibles, les outils de conception et les algorithmes

- architectures générales SIMD:

  compilation parallèle grain fin, réseau d'interconnexion, modèle de programmation SPMD, langages data-parallèles, algorithmique SIMD

- architectures générales MIMD:

  processeurs de communication, hiérarchie mémoire, noeud hétérogene de calcul et de communication, réseau d'interconnexion, modèles d'exécutions et lien entre processus communicants et programmation SPMD, compilation parallèle gros grain, algorithmique MIMD, outils de communication dans les réseaux, outils de placement et d'ordonnancement, modèles de machines virtuelles.

Si chacun des points cités ci-dessus doit faire l'objet de recherches précises, il semble important aujourd'hui pour aller plus loin de mettre l'accent sur les relations entre les langages capables d'exprimer le parallélisme massif, les modèles d'exécution et les architectures matérielles et logicielles associés.

Ces relations portent notamment sur les points suivants :

1. Les modèles d'exécution et leur relation avec les langages et les architectures

   Les modèles d'exécution dynamiques sont fondés sur l'interprétation. C'est le cas des processus communicant par passage de message dans les machines à base de transputers ou les hypercubes. Le modèle SPMD conserve des communications interpretées, par passage de messages. La compilation des communications, pour les applications prévisible à la compilation, essaie d'étendre aux machines massivement parallèles le rapport d'efficacité entre approche compilée et approche interprétée des machines séquentielles. L'efficacité des différents modèles est à mettre en relation avec le type de langage utilisée (data-parallèle ou expression de la concurrence) et avec les rapports calcul-communication dans les architectures matérielles.

2. La parallélisation automatique gros grain et son influence sur les architectures matérielles et les langages.

   Les techniques de parallélisation automatique ont progressé avec notamment dans la dernière période l'apport des techniques de systolisation. Les relations entre ces techniques et l'architecture matérielle sont importantes des deux points de vue:

   - techniques de compilation adaptées en fonction des architectures matérielles utilisées, et
   - définition de nouvelles architectures matérielles adaptées aux techniques de compilation les plus avancées.

   Les structures matérielles mixtes combinant les approches parallèles et vectorielles sont-elles viables ? Quelle est l'influence des progrès en architectures matérielles et en compilation sur l'évolution des langages parallèles : de Fortran 90 vers quel langage ?

3. La compilation parallèle grain fin et son intéraction avec la structure des processeurs.

   Comment articuler efficacement le microparallélisme (pipeline et superscalaire) des processeurs élémentaires avec le parallélisme de l'application ?

4. Les communications

   Comment accélérer les communications, d'un point de vue matériel (réseaux d'interconnexion performants) et logiciel (compilation ?) ? Les intéractions entre les communications et

   - la distribution automatique des données

     L'utilisation de mémoires distribuées soulève un certain nombre de questions fondamentales, car elle implique un problème délicat de répartition des données

     - efficacité du partitionnement des données en fonction des relations calcul, communication, débit mémoire
     - impact de la hiérarchie mémoire utilisée au niveau de l'architecture matérielle (caches associés aux processeurs RISC) ou pour faciliter l'exploitation (mécanismes de mémoire virtuelle).

   - les langages data-parallèles
   - les systèmes d'exploitation

     partitionnement et ordonnancement, mémoire virtuelle distribuée
   - les réseaux sophistiqués

     Quel est l'apport réel des réseaux recombinants, à routage automatique, gérant directement les tableaux, etc ?

Ce thème de l'appel d'offres met l'accent sur l'étude des relations "langages-modèles d'exécution-architectures" pour le parallélisme massif. C'est pourquoi il est sollicité la constitution d'un projet permettant d'étudier de manière précise et approfondie ces relations.

L'accent doit être mis sur l'analyse quantitative, en s'appuyant sur la réalisation de maquettes matérielles, de maquettes logicielles, et sur des mesures sur des plates formes d'expérimentation (machines existantes et prototypes).

ENSEIGNER ADA

Michel GAUTHIER
maître de conférences
animateur de Ada-France
Laboratoire d'Informatique
faculté des sciences de Limoges
123, avenue Albert Thomas
F-87060 Limoges cedex
téléphone **(33) 55 45 73 35**
télécopieur **(33) 55 45 72 01**
gauthier@unilim.fr

Ada-France
groupe «Ada» de l'AFCET

à
correspondants de SPECIF
et
enseignants membres de Ada-France
et
tous enseignants intéressés à Ada

Lettre à copier et faire circuler si vous l'estimez utile.

Limoges, le **7 juillet 1992**

Cher collègue,

Lors de rencontres informelles avec divers collègues, il a été constaté un besoin de connaissance de l'existant et d'échange d'expériences entre les universitaires s'intéressant à Ada.

Deux associations professionnelles, l'*AFCET* —dont *Ada-France* est un groupe de travail— et *SPECIF*, sont a priori compétentes pour organiser de telles rencontres. Animateur de la première et correspondant local de la seconde, je me permets de faire quelques propositions. Si vous n'êtes vous-même pas le bon destinataire dans votre université, je vous demanderai de transmettre ce courrier à tout collègue susceptible d'être intéressé.

Une première tâche pourrait être un recensement des enseignements sur, autour de, ou utilisant Ada, dans les établissements d'enseignement supérieur public en France (divers cycles des universités, écoles d'ingénieurs, IUT,...). Une liste de ces enseignements, avec pour chacun un point de contact, pourrait être diffusée par des voies à déterminer.

S'agissant des contacts, une difficulté permanente est de savoir comment diffuser l'information. Ceux qui suggèrent l'emploi de tel ou tel système électronique doivent être conscient qu'aucun n'est universel, et que tous les collègues n'y ont pas nécessairement accès. S'il est vrai qu'un dépôt d'informations en ASCII pur peut être installé sur le serveur du CNAM, ce n'est pas encore la solution universellement accessible.

J'émettrai de plus l'hypothèse qu'en de nombreux cas, les destinataires procéderont à une copie sur papier. Ceci me conduit finalement à proposer

la création d'un bulletin de liaison, co-parrainé par *Ada-France* et *SPECIF*, et dont la réalisation pourrait être assurée à tour de rôle par les divers établissements intéressés. Ceci me semble la solution la plus simple, qui simplifierait le fonctionnement en éliminant tout recours à des transferts d'argent (le troc serait-il une idée moderne?).

Je propose donc à tous les intéressés de me passer un petit mot (papier de préférence, ou courrier électronique) avec toutes les informations qui leur semblent utiles. La liste complète de ces expéditeurs leur sera expédiée, mais ce qui se passera ensuite dépendra bien entendu des réponses que je recevrai. Si l'idée d'un bulletin vous agrée, j'aimerais en particulier quelques engagements à assurer la réalisation et l'envoi (ça ce n'est pas cher...) d'au moins un bulletin. Il faudrait aussi un petit groupe de direction (j'ai déjà quelques contacts, et je ne tiens pas spécialement à m'ajouter du travail supplémentaire).

J'attends vos idées et vos réactions.

Par ailleurs, un groupe de travail est en cours de création à *Ada-Europe*, dans le but de collecter, et éventuellement de solliciter, des composants Ada spécifiquement destinés à l'enseignement. D'autres informations suivront, mais vos réactions immédiates pourraient m'être utiles. La première initiative serait de demander aux enseignants quels sont les trois composants, ou éventuellement sous-systèmes, qui leur sembleraient les plus utiles pour l'enseignement.

Bien confraternellement à tous.

Enseigner Ada ?

Lorsque Daniel Feneuille m'a demandé quelques mots pour accompagner son texte sur l'expérience d'enseignement de Ada dans les I.U.T, ma première réaction, pas très raisonnable à la réflexion, fut de me dire que les bons produits n'avaient pas besoin de publicité, et que donc ni lui ni Ada n'avaient besoin de mon apport supplémentaire. Et puis, après tout, pourquoi ne pas en profiter pour poser quelques problèmes? Et n'avais-je pas tort de penser qu'une information à caractère non publicitaire était devenue superflue ?

J'ai commencé à enseigner Ada à l'université en licence de mathématiques en 1988, dès que nous avons reçu notre Vax (seule possibilité à l'époque d'avoir un compilateur Ada dans une petite université), après toutefois quelques formations continues aux Télécom. Il s'agissait d'une formation en direction d'étudiants ayant eu un premier cycle renforcé, en raison essentiellement de l'absence de licence d'informatique à cette époque. L'expérience principale que je tirai de cet enseignement fut que la principale difficulté provenait plus du changement d'un langage pour un autre que de difficultés propres à Ada, et une autre du manque de temps à consacrer à un module en définitive mineur du diplôme.

De là à réfléchir sur un possible enseignement de Ada à des débutants, il n'y avait qu'un pas, que je franchis au printemps 1989. Je suis évidemment heureux d'apprendre que cela a pu servir à quelque chose.

Pourtant, enseigner Ada, et tout particulièrement à des débutants, pose quelques problèmes bien identifiés :

- les choix didactiques utilisés pour les autres langages risquent d'être inapplicables,
- les exercices habituellement utilisés ne sont plus nécessairement adaptés,
- les enseignants qualifiés n'étaient pas légion à cette époque,
- divers composants doivent être conçus au préalable, et fournis aux étudiants,
- les compilateurs ne sont pas toujours au niveau de qualité des messages d'erreur nécessaire pour des débutants,
- il existe une opposition active à ce langage au sein des universités.

L'expérience se devait donc d'être faite, et le papier sur la rencontre des I.U.T. en fait un bilan utile. Au moins les trois premiers points commencent à trouver leur solution.

Il reste donc trois difficultés, sur lesquelles je crois utile d'insister.

Commençons par les compilateurs. L'écoute des utilisateurs tend à établir le classement de qualité suivant : Rational (mais peu d'universités en possèdent), Digital, et tous les autres ex-æquo, sauf en dernier Meridian. C'est un problème réel, car un mauvais compilateur peut dégoûter un utilisateur. N'ayant d'intérêt personnel nulle part, je me permettrai un avis clair : évitez Meridian, et si possible prenez Digital ou mieux encore Rational. Avec les conditions Logidec du marché entre l'Education Nationale et Digital, tout centre de calcul universitaire peut fournir un accès au meilleur des possibilités de Ada. Si vous devez prendre l'un des compilateurs classiques du monde Unix, soyez exigeants auprès des constructeurs quant à la prise en compte des rapports d'anomalies.

Cela dit, si le problème de la qualité des compilateurs est réel, et s'il complique l'enseignement, les qualités pédagogiques de Ada compensent amplement cette difficulté par rapport à des langages plus faciles à compiler, mais souvent peu portables, et exigeant de l'utilisateur une adaptation à des détails d'inutilement bas niveau.

La question des composants pour l'enseignement est importante, et ne peut pas trouver de solution dans une action individuelle ou locale. Elle appelle une prise en

compte collective. C'est l'idée que j'avais en tête lorsqu'au printemps dernier j'ai proposé à Ada-Europe la constitution d'un groupe de collection de tels composants. C'est la même idée qui m'a fait écrire un peu plus tard la lettre (jointe à ce papier) aux correspondants de SPECIF, afin de construire un tel groupe aussi en France.

Je n'ai à ce jour que peu de réponses, mais peut-être la sortie de ce bulletin m'en vaudra-t-elle d'autres. Et j'insisterai sur ce premier sondage d'opinion : quels sont les trois composants (ou éventuellement sous-systèmes) qui vous paraissent les plus importants à fournir aux étudiants ?

Quelques mots enfin sur les oppositions à Ada en milieu universitaire. Certains m'ont dit leur aversion pour son origine militaire, je n'essaierai pas de les convaincre, mais je vois dans Ada surtout un cahier des charges et une définition issus d'une discussion générale comme l'informatique en a fort peu connues.

Les oppositions sérieuses sont ailleurs, dans certains autres langages et leurs concepts associés, allons-y clairement et donnons des noms : C, le monde Lisp, Prolog, le monde des objets. Je refuse personnellement toute discussion qui s'appuie sur des considérations de chapelle[1], elles ne m'intéressent pas. Je refuse tout débat qui ignorerait la nécessité de former les étudiants au doute et à la capacité critique, et donc à la compréhension des rapports des langages de programmation entre eux dans leur lutte pour la survie.

Mais ceci ne vaut pas pour les débutants, pour lesquels il semble préférable qu'un langage unique soit choisi. J'ai écrit précédemment dans ce même bulletin ma conviction qu'il fallait tuer Pascal, et je le confirme. Les qualités pédagogiques habituellement reconnues dans Pascal se retrouvent intactes, voire améliorées, dans Ada, sans inconvénient associé. En plus, le mécanisme d'exception constitue un apport essentiel pour l'enseignement, après des années passées dans un mutisme gêné dès qu'il fallait exprimer qu'une opération était impossible. Les travaux de la rencontre des I.U.T. confirment la possibilité et l'intérêt de ce remplacement.

Je ne crois pas vraiment à l'efficacité d'enseigner à des débutants C, Lisp ou Prolog, langages où l'on doit toujours se préoccuper du comment-ça-marche, et finalement d'une plus grande complexité que Ada. Cela dit, et parce que je ne le crois pas, je suis à l'écoute des arguments et expériences de ceux qui l'auraient tenté. Qu'ils écrivent !

J'ai déjà eu l'occasion de dire mon opinion sur le débat entre Ada et les objets, mais il peut être judicieux de la rappeler. L'intérêt d'une *conception fondée sur la réalité* est indéniable, et il est souhaitable que les objets de la réalité trouvent dans les structures des langages les moyens de leur modélisation. Ada et les langages à objets constituent deux réponses possibles. Au-delà de cette idée commune, la divergence apparaît, concernant notamment l'importance et le statut de l'héritage (concept ou seulement outil de certains langages ?) et l'universalité du concept d'objet (y a-t-il autre chose que des objets ?). Au fond, il semble qu'il s'agisse de deux démarches opposées, généralisation précoce pour Ada et adaptation à la demande pour les langages à objets, et qui restent à ce jour incompatibles, peut-être comme la relativité générale et la mécanique quantique. Les deux possibilités sont à mon sens cohérentes, mais il est inévitable d'en choisir une.

A ce niveau, la décision relève de l'histoire individuelle de chaque enseignant, et l'énorme acquis théorique et conceptuel autour des langages à la Pascal est un avantage important en faveur de l'enseignement de Ada.

Michel GAUTHIER

---

[1] Un paranoïaque, c'est une personne qui croit à tort que les autres lui veulent du mal lorsqu'ils affirment avec raison qu'il l'est.

ADA et les I.U.T.

Les 21 et 22 Mai 1992 se sont tenues à DIJON les journées "Enseignement de ADA dans les I.U.T. Informatique". Placée sous le présidence de Jean ICHBIAH qui présenta deux exposés, ces deux journées ont reçu les communications des départements de DIJON, ORSAY, VELISY, GRENOBLE et AIX, départements qui ont fait part de leur expérience plus ou moins confirmée de leur enseignement de ADA.

Pour ces journées nous avions, à AIX, proposé le papier qui va suivre. Ces dix pages, N. COT nous a suggéré de les publier pour SPECIF. Nous ne les avons presque pas modifiées. Mais il va sans dire que nos réflexions concernent avant tout les départements informatiques des I.U.T. donc concernent des enseignements très professionnalisés à hauteur de 700 heures d'informatique sur deux années. Le lecteur intéressé y trouvera les arguments pédagogiques qui nous ont fait opter (sans retour!) pour ADA; citons rapidement:

fort typage et contrôle sévère à la compilation,
bon traitement des anomalies,
nécessité, ou tout au moins possibilité, de beaucoup spécifier,
implémentation assez agréable de types abstraits,
réutilisation facilitée, notamment par la généricité,
portabilité (enfin!),
tâches et synchronisation très "propres",
.....

Certes tout n'est pas idillique et si nous avons encore des griefs, nous faisons nôtre l'aphorisme (que l'on doit sûrement à J.P. ROSEN):
"Ceux qui se sont mis à ADA passent leur temps à énumérer leurs récriminations mais pas un ne retournerait à son ancien langage".

Notre expérience est-elle transposable? Nos réflexions peuvent-elles intéresser d'autres cursus? Tel est peut-être l'intérêt de notre document.

M. GAUTHIER, dans une publication interne du L.R.D.I. de Limoges datée de 1989, dit sa conviction qu'ADA est aussi un bon choix pour des débutants qui ne feront pas d'informatique poussée. ("A propos de la didactique particulière de ADA"). Ce papier avait levé, à l'époque, nos dernières appréhensions. Notre collègue est, à l'heure actuelle, chargé par ADA-FRANCE d'une mission de prospective sur ADA et son enseignement, mission à laquelle il doit associer SPECIF. Nous ne saurions que recommander aux collègues intéressés de répondre à son appel pour former un groupe de réflexions, d'échanges et de diffusions.

D. Feneuille Aix Septembre 1992

PS: Nous pensons être en mesure, dans deux ans environ, de dresser un bilan plus complet et montrer notre évolution vraisemblable.

ADA : épine dorsale de l'enseignement de l'Informatique à l'I.U.T d'Aix en Provence

(Journées "L'enseignement de ADA dans le D.U.T. Informatique" DIJON 92)

T. Avignon, D. Feneuille, D. Mathieu

I - Bref historique

Pour situer la problématique et pour commencer, quelques dates approximatives ayant marqué des choix importants dans notre enseignement :

1975-1976 : abandon des organigrammes au profit de la programmation structurée -langages enseignés : COBOL, BASIC, FORTRAN, GAP (dur de faire propre!)
1981-1982 : équipement du département "tout-micro" - choix du langage PASCAL et abandon progressif du FORTRAN (GAP et BASIC ont disparu sans qu'on s'en aperçoive ...)
1983-1984 : enseignement du langage C et d'UNIX
1989-1990 : renouvellement du matériel en stations de travail - premier enseignement d'ADA,
1991-1992 : Suppression du COBOL et de PASCAL, remplacés (en nombre d'heures) par ADA, notoire diminution de l'assembleur, "remplacement" de C par C++

II - Quelques principes

1 - Dès la création des IUT, leur vocation fut de former des techniciens supérieurs capables de réaliser des travaux concrets, en étant opérationnels, efficaces et méthodiques - des **pro's** - (d'où la nécessité de leur faire acquérir un savoir-faire et de la (ou des) méthodes), et capables aussi de comprendre les ingénieurs et de travailler avec eux (nous y reviendrons).

2 - De tout temps, nous n'avons pas considéré les langages de programmation comme une fin en soi, mais comme le moyen - éphémère - de faire passer de façon concrète des concepts fondamentaux et (presque) éternels !!!

3 - Nous pensons enfin qu'un bon enseignant n'est là ni pour se faire plaisir, ni pour se la couler douce. Cela signifie qu'il accepte de remettre cent fois sur le métier son ouvrage, et de remettre en cause son enseignement bien rodé au profit d'un autre, qui sent parfois encore la peinture fraîche, mais qui a l'air plus prometteur.

III - Avant ADA

Comme tout un chacun, nous avions organisé l'enseignement de l'informatique en modules. A chacun d'eux correspondait un "langage" privilégié servant à mettre en oeuvre les notions étudiées dans le module :

Algorithmique	PASCAL
Récursivité	PASCAL
Structures des machines	Assembleur 8086
Arithmétique (calculs numériques)	PASCAL
Fichiers (organisation et cinématique)	COBOL
Structures de données	PASCAL
Evaluation - Coûts d'algorithmes - Tris	PASCAL
Systèmes d'exploitation	C - UNIX - DOS
Parallélisme	C

IV - Les frustrations avec ces langages

Commençant notre enseignement de l'informatique, bien évidemment, par l'algorithmique, nous n'avons jamais voulu utiliser le langage C comme langage "support"... Dès lors il ne nous restait plus qu'à enseigner PASCAL! Voici pourtant quelques unes des insatisfactions qui s'accumulèrent au bout de quelques années :

PASCAL

Fervents pascaliens de longue date, par sa connexité avec l'algorithmique, nous avions cependant quelques griefs à son égard :

- impossibilité de séparer les interfaces des implémentations,
- impossibilité de masquer la structure des données en ne fournissant que les "méthodes" (En ce temps là le Pascal-objet n'était pas répandu !!!),
- plus fondamentalement, grosses difficultés à concevoir et surtout à mettre en oeuvre des traitements d'erreurs satisfaisants : erreurs exceptionnelles, points de reprise dans les applications, sorties en "catastrophe",
- indépendance des traitements généraux vis-à-vis de la nature des données traitées (par l'absence de généricité) obtenue au prix de mille contorsions !
- des détails "pas propres" et sources d'erreurs : passage des paramètres par référence pour gagner de la place (vecteurs), pas de paramètres "données-résultats",
- sensation de toujours recommencer le même code à l'infini pour effectuer des traitements identiques (listes d'entiers, de réels, de chaînes, de ...),
- nécessité d'avoir des quantités d'identificateurs de fonctions différents désignant des fonctionnalités identiques appliquées à des types différents (absence de surcharge), comme par exemple x^y avec x et y de différents types (entiers, réels, octets, etc..),
- absence d'entrées/sorties sur types énumérés, qui en rend l'utilité assez réduite,
- non-portabilité des algorithmes numériques,
- problèmes de visibilité (indices de boucles For pas satisfaisants), gros problèmes en cas de déclarations multiples d'un même identificateur dans des unités différentes,
- passage de procédures en paramètres impossible,
- pas de notion de processus
- bogues découverts trop tard dans la mise au point
- typage trop faible, par exemple pour les variables numériques

Quant aux autres:

Assembleur : il nous paraît qu'il y a une masse critique du nombre d'heures d'enseignement. Au delà de ce nombre, on peut supposer que les étudiants pourront se débrouiller professionnellement. En deçà, il n'est plus question de développer des applications intéressantes, et il n'y a plus qu'à réduire le volume de façon drastique, ne gardant que les notions fondamentales, et s'appuyant sur un langage plus évolué (PASCAL, ADA ou C).

COBOL : gag !!!

Langage C : malgré de nombreux avantages, le nombre de pièges qu'il renferme est le plus important inconvénient. Les programmes des étudiants sont de véritables nids de vermine dont il est difficile de sortir. Il est difficile d'avoir confiance dans un programme C, à moins d'avoir une très grande pratique de ce langage, ce qu'il est matériellement impossible de faire acquérir aux étudiants.

On remarque que ne figurent pas les enseignements suivants : Réseaux, Intelligence artificielle, A.C.S.I., Bases de données. La raison est qu'ils sont effectués par des enseignants non encore convertis, mais çà viendra sûrement!

V - Et aujourd'hui (avec ADA)

Organisation générale - volumes des modules - notions - place de ADA:

Comme un bon dessin vaut souvent mieux qu'un long développement nous vous invitons à survoler les deux grilles-planning de notre enseignement de première année sur 32 semaines (cf annexes). Dans ces documents on voit (en hachuré et en grisé) les modules où ADA intervient soit comme composant essentiel soit comme vecteur d'enseignement. En regard de cette grille, semaine après semaine, on trouve les concepts ADA introduits. La séquentialité de ces concepts peut laisser supposer une relation d'ordre strict entre ceux-ci. Il n'en est malheureusement rien car ADA, nous semble-t-il, est bien plus qu'un langage de programmation. Et c'est bien là une des difficultés de son enseignement où il faudrait tout faire (ou presque) en même temps pour montrer sa grande cohérence. Il nous arrive parfois, de façon transparente, d'introduire par exemple généricité et exception (notions difficiles mais combien essentielles) dans les premiers travaux confiés aux étudiants. A charge, bien sûr, d'y revenir de façon plus approfondie au moment opportun.

ADA et les modules concernés:

Nous allons succinctement passer en revue les modules avec lesquels ADA cohabitent. Une place à part , et plus importante, est faite au module Système d'exploitation (enseigné en deuxième année).

ADA et module "Algorithmique" :

Dans ce module, qu'il n'est pas besoin de détailler tant son contenu est homogène quels que soient les départements Informatiques des I.U.T. qui l'enseignent, nous retiendrons pour ADA quelques avantages :

- très fort typage qui ne surprend pas les vrais débutants et permet de ramener les étudiants initiés à des pratiques saines de programmation propre et rigoureuse où de nombreux bogues sont mis à jour à la compilation.
- existence d'une sortie de boucle (`exit`) qui évite beaucoup de contorsions algorithmiques.
- utilisation, déjà, d'exceptions pour gérer simplement les erreurs rédhibitoires, par exemple l'initialisation de variables saisies au clavier
- première utilisation simple d'attributs (opérations sur un type)
- transformation facile d'une constante chaîne en une donnée de type énumératif, et réciproquement, grâce aux attributs.

ADA et Coûts d'algorithme"

Dans ce module, nous avons cherché à faire réfléchir les étudiants sur les "coûts" d'algorithmes différents résolvant parfaitement un même problème. L'un des thèmes de ce module, central, est le tri, notamment de vecteurs. Nous avons là de quoi illustrer amplement les notions de sous-programmes, ainsi qu'une nouvelle notion introduite par ADA : les tableaux non contraints. Quoi de plus facile que d'introduire la notion de **modèle de type tableau** (`array range <>`). Ces modèles nous servent pour déclarer et définir les paramètres formels des sous-programmes, les paramètres réels étant, eux, définis à partir de **sous-types contraints** de ces modèles de types.

Quant aux sous-programmes, ils ne peuvent être écrits que grâce aux attributs des types discrets (`FIRST`, `LAST`, `RANGE` et `LENGTH`).

ADA et récursivité

ADA : épine dorsale de l'enseignement de l'Informatique à l'I.U.T d'Aix en Provence
(mai 1992)

Avec ADA, nous avons pu sans difficulté reprendre tous nos exercices écrits en PASCAL pour illustrer notre cours sur les différents types de récursivité. A noter que ADA permet, grâce aux exceptions, de sortir de façon élégante mais brutale, du plus profond des appels récursifs.

ADA et arithmétique

On peut enfin et vraiment parler de **portabilité** sans se soucier des caractéristiques des machines qui réaliseront les algorithmes programmés. Nos étudiants passent sans problème des P.C. aux DEC (et réciproquement).

Pour le type entier : l'utilisation systématique de types formellement définis comme **dérivés** d'un type entier standard (mais que l'on ne nomme pas et qui est choisi par le compilateur grâce à la déclaration de contrainte d'intervalle) résout parfaitement la portabilité annoncée.

Pour le type réel : grâce à la déclaration `digits` et sans se soucier des implémentations (notion de **nombre sûr**), l'utilisateur définit les caractéristiques de ses nombres personnels (notion de **nombre modèle**) et ADA s'engage à réaliser au moins ce contrat (mantisse) ou alors renonce à la compilation. Les calculs réalisés sont enfin à défaut d'être identiques, sur des machines différentes, conformes à la précision demandée.

Enfin grâce aux nombres **"delta"** pour une certaine plage de contraintes, on peut utiliser des réels où le "trou" entre deux éléments contigus, sera constant en valeur absolue (intéressant pour des calculs financiers enseignés dans le module gestion).

ADA et Structures des données (Pile, liste et arbre)

Ce module utilise et concrétise simultanément les notions suivantes, très importantes et difficilement dissociables :

- type enregistrement avec discriminant
- paquetage et types privés
- généricité
- exceptions

En introduction, nous utilisons la chronologie suivante, entièrement fondée sur l'utilisation de la notion de pile :

1 - pile d'entiers construite à l'aide d'un paquetage. La structure de la pile est entièrement encapsulée dans le corps. Spécifications et corps sont toujours séparés.

2 - création d'un nouveau paquetage fournissant un **type T_PILE privé** garantissant toujours la confidentialité de la structure de pile.

3 - création d'un nouveau paquetage fournissant un type privé avec discriminant, permettant de manipuler des objets de types différents, mais prévus dès la conception (entiers, réels). Cet exemple nous permet d'aborder le **polymorphisme**.

4 - création d'un paquetage **générique** dont le paramètre de généricité est le type de l'objet à empiler.

C'est à ce niveau de l'enseignement que l'on commence à faire un large usage des exceptions pour valider **toutes** les opérations.

Dès que toutes ces notions sont acquises, nous pouvons bâtir des applications consistantes mettant en oeuvre les structures de plus en plus compliquées : listes simplement puis doublement chaînées, listes triées, arbres binaires, arbres de recherche, arbres équilibrés etc... Nous consacrons un temps non négligeable à l'élaboration et à l'assemblage des spécifications avant de passer au codage des algorithmes proprement dits. **Tous** les paquetages sont génériques, de façon

à bien dissocier les structures elles-mêmes des informations qu'elles stockent.

ADA et assembleur

Nous reprenons ici sans originalité ce que nous arrivions à faire avec PASCAL, à savoir utiliser en ADA des sous-programmes (même paramétrés) écrits en langage machine. Ceci permet simplement de rendre le module assembleur solidaire des autres enseignements et de récupérer en ADA de temps à autre, quelques algorithmes de très bas niveau réalisés en assembleur (8086). Cette récupération est rendue possible grâce à la directive de compilation `PRAGMA INTERFACE`.

ADA et fichiers

Alors que les notions de fichiers (texte, séquentiel et direct) étaient introduites assez rapidement avec PASCAL et COBOL, nous avons volontairement reporté l'étude de ce chapitre après le cours de structures de données pour plusieurs raisons :

- la première est que, pour manipuler les fichiers textes, il nous faut utiliser le paquetage TEXT_IO qui contient lui-même des paquetages génériques (INTEGER_IO, FLOAT_IO, ENUMERATION_IO, DELTA_IO) à instancier. Pour les fichiers séquentiels et les fichier directs, il faut instancier les paquetages SEQUENTIAL_IO et DIRECT_IO. Les étudiants ne peuvent utiliser pleinement et systématiquement ces paquetages qu'après avoir acquis la généricité.

Il y a donc lieu dans toute la première partie de notre enseignement, de mettre à la disposition des étudiants un paquetage "maison", facile à utiliser, permettant d'effectuer les entrées/sorties simples et transparentes à partir du clavier et de l'écran.

- la deuxième est que nous construisons nos programmes comme nous l'avons fait avec les structures de données. En effet comme l'indique Grady Booch, il est possible de voir un fichier séquentiel indexé comme un objet se décomposant lui-même en un objet *fichier direct* et un objet *index* (table d'index). Après avoir réalisé un paquetage générique de gestion de fichier séquentiel indexé s'appuyant lui-même sur un paquetage générique de gestion d'un fichier direct et d'un paquetage générique de gestion d'une table d'index, il nous est possible de concrétiser la notion de **réutilisabilité** en remplaçant le paquetage de gestion de la table d'index par le paquetage générique de gestion de l'arbre de recherche réalisé dans le module "Structures de données".

ADA et module "Système d'exploitation"

Le module "Système d'exploitation" représente un volume de 90 h réparties sur les deux années. Il est constitué (en 91-92) de 40h de cours en amphi et de 50 h de TD-TP. Avant de parler des apports d'ADA, précisons rapidement le plan de cours :

I - Introduction
II - Qu'est-ce qu'un processus
III - Gestion des activités parallèles
IV - Gestion mémoire
V - Gestion des processus
VI - Gestion des fichiers

Le chapitre 3 (14 h de cours et 28h de TD-TP) est lui-même structuré de la façon suivante :

1 - Définitions - Parallélisme - Ressources - Exclusion mutuelle
2 - Synchronisation :
  - les sémaphores
  - les moniteurs
  - les rendez-vous d'ADA

3 - Communication entre processus
- les signaux, les pipes et les sockets sous UNIX
- les rendez-vous d'ADA

Comme on le voit, (C + UNIX) d'une part, ADA d'autre part offrent tous les outils permettant de mettre en oeuvre concrètement des activités parallèles. Si on y joint un peu de réseau (lien entre le cours réseau TCP/IP et la communication par sockets citée ci-dessus), on a même pu faire concevoir et réaliser aux étudiants des applications réparties sur plusieurs sites différents.

Le titre du chapitre **"Gestion des activités parallèles"** suppose explicitement qu'on soit capable de créer des activités parallèles. Il suppose aussi implicitement qu'on soit capable de découper un problème en activités suffisamment autonomes pour être parallélisables. Il s'agit donc dans un premier temps de montrer aux étudiants que des applications importantes sont plus faciles à appréhender et à concevoir si on arrive à isoler des activités indépendantes. Les **tâches en ADA** et les processus en C sont tout-à-fait ce qu'il nous faut. L'avantage d'ADA, en ce qui nous concerne, est que l'investissement complémentaire en syntaxe par rapport au niveau acquis jusque là est très minime : `task` et `task body`, qui est tellement simple à partir de `package` et de `package body`. Il est donc extrèmement simple de faire faire des tâches à des étudiants. Dès lors les évènements se précipitent :

- la synchronisation de deux activités A et B devient nécessaire : syntaxe très simple : `accept RDV`; la mise en oeuvre est plus simple et plus propre que signal du C

- les activités A et B ont "envie" de communiquer des informations en même temps qu'elles se synchronisent : `accept RDV (xxx);`

- les activités A et B ont "envie" de pouvoir être "découplées", par l'utilisation d'une boite aux lettres : A confie l'information à une tierce tâche S auprès de laquelle B la retire. L'exclusion mutuelle est introduite au passage "comme une lettre à la poste"... De plus la notion de client (tâches A et B) et de serveur (tâche S) apparaît toute naturelle.

- il ne reste plus qu'à compliquer de façon très progressive et très naturelle : le serveur se met en attente sur plusieurs services : attente sélective; c'est l'instruction `select accept ..` Il soumet les services qu'il peut rendre à des contraintes temporelles : (délai d'attente `delay`), ou à des conditions (existence de ressources, etc...) et nous revoilà dans les systèmes d'exploitation !!! C'est l'attente gardée `when condition => accept ...`

- les clients deviennent exigeants et soumettent leurs demandes à des contraintes de délai: s'ils ne sont pas servis à temps, ils s'en vont ....

Il ne reste plus qu'à introduire une notion qui nous a beaucoup séduits en ADA : le **type tâche** `type task`. D'un aspect anodin, cette notion va offrir de nombreuses possibilités :

- on peut créer plusieurs tâches strictement identiques (exemple classique des sémaphores : un sémaphore = une tâche, mais est-ce bien raisonnable ?).

- on peut créer (et détruire) des tâches de façon dynamique, simplement - comme de vulgaires données classiques - avec un pointeur dessus. Dès lors, les problèmes de chiens en nombre quelconque qui se courent après deviennent presque des jeux d'enfants. La notion d'objet prend une dimension nouvelle : chaque tâche a sa partie cachée (ses données et ses fonctions/procédures locales), ses procédures visibles (les `entry's`) qui permettent de communiquer avec elle. Leur comportement est celui prévu par leur type, et, avantage par rapport à leurs cousins de C++, ces objets ont une "vie" autonome et parallèle.

- et comme l'enseignement fait un tout, il est très agréable de pouvoir réutiliser des paquetages déjà écrits. Nous apprécions de nouveau toute la puissance de la généricité en instanciant les paquetages développés pendant le cours sur les structures de données (listes et

arbres) avec des tâches : quel autre langage permet de faire des listes de tâches, des arbres de tâches, sans écrire une ligne de code, et en garantissant que "cela marche du premier coup" ?

Pour en finir, la transition entre un serveur en ADA et un serveur en C communiquant au moyen de sockets est très facile. Il existe en effet certaines analogies dans la structure des programmes et dans les instructions elles-mêmes.

VI - Quelques réserves cependant!

Il ne s'agit pas ici pour nous de critiquer le langage ADA en lui-même. Nous ne nous le permettrions pas ... Nous avons cependant éprouvé quelques gênes que nous exposons ici, en béotiens :

- une bonne partie de notre matériel est constitué de PC, pour certains assez anciens, et de capacité mémoire limitée. de ce fait nous arrivons assez vite à saturation, après instanciation de 2 ou 3 paquetages,

- ayant l'habitude d'utiliser les pointeurs de fonctions en C, nous sommes très gènés par l'impossibilité de passer des fonctions en paramètres autrement que par la généricité,

- la communication entre processus (ou tâches) semble beaucoup moins souple qu'en C sous UNIX, en particulier entre machines distantes. Il manque des outils tels les pipes ou les sockets, etc...

- nous ne disposons pas de paquetages graphiques efficaces sur PC, pouvant rivaliser avec ceux de BORLAND . De ce fait, la convivialité des applications développées par les étudiants, et la qualité de la présentation des résultats reste toujours très insuffisante,

- nous n'avons pas évité l'écueil bien connu de la coexistence très moyenne entre ADA et UNIX. Le blocage de toutes les tâches par une seule qui demande des E/S est toujours très pénalisant (cette remarque n'est plus d'actualité en septembre 92), résolu par différents moyens: certains compilateurs, boîtes à outils (paquetage "paradise"), ...,

- la remontée des procédures et fonctions d'une couche logicielle à une autre par `renames` est fastidieuse et nous fait regretter l'absence d'un mécanisme d'héritage présent dans les langages à objets.

VII - Passage à C++

L'introduction du langage C++ dès la fin de la première année arrive à point nommé pour contourner quelques difficultés. L'acquisition par les étudiants, au cours des six premiers mois, des notions d'objets (ou tout au moins de type abstrait), d'encapsulation, d'abstraction, présentes dans ADA, leur permet une adaptation très rapide à un langage de programmation par objets. Tout le début de l'enseignement de C++ est fondé sur la comparaison des deux langages (ressemblances et différences des syntaxes, mais surtout des concepts). Les classes sont introduites au bout de quelques heures d'enseignement, sans difficultés particulières pour nos étudiants.

VIII - Conclusions

- A ceux qui nous disent : ADA n'est pas encore assez implanté dans les entreprises, nous répondons : et alors ? Devons-nous être en retard sur le milieu professionnel ? D'ailleurs, qui peut dire pour combien de temps ? Et l'Université, à laquelle nous appartenons, n'a-t-elle pas aussi pour rôle de "produire" des jeunes professionnels qui enrichiront par leur diversité la culture des équipes d'informaticiens des entreprises ? Par contre ce qui est clair à l'heure actuelle c'est que nous avons bien du mal à trouver des stages, de fin de scolarité pour nos étudiants, en ADA!

- D'autres prétendent que c'est d'un trop haut niveau pour des techniciens supérieurs. Alors à

qui seraient-ils supérieurs ? Un snobisme teinté de mépris de quelques collègues universitaires les porte à croire que les "morceaux nobles" sont réservés aux ingénieurs (Génie logiciel, Langages à objets, ADA, I.H.M, etc...) tandis que le Basic, le COBOL etc ... sont bien assez bons pour nos modestes techniciens! On imagine bien à court terme la cohérence de telles équipes ingénieurs-techniciens! Il faudrait alors peu de temps pour donner raison à ceux qui pensent qu'en informatique, il n'y a besoin **que** d'ingénieurs.

Nous avons essayé de montrer en quelques pages la cohérence de notre démarche pédagogique. Pour résumer, nous avons voulu faire passer le maximum de concepts pour un investissement "minimal" en langage. Il est évident que la syntaxe de l'ADA n'est pas minimale en soi, mais le fait de couvrir le spectre le plus large possible de problèmes avec un langage unique permet d'aller beaucoup plus loin, de moins se disperser. Il n'est pas dans notre intention de dire qu'ADA est la panacée - d'ailleurs nous n'en avons pas la compétence - ni de montrer que notre démarche est la seule possible et la meilleure : d'autres parmi vous ont pris ou prendront d'autres voies. Nous avons voulu seulement vous faire partager notre expérience, montrer qu'elle est possible, viable, et qu'elle satisfait à notre cahier des charges. Elle n'est pas achevée, loin de là, mais elle s'appuie déjà sur trois années d'expériences.

Nous pensons enfin pouvoir affirmer les deux points suivants (en guise de mise en garde) :

- l'apprentissage d'ADA n'est pas possible à petite dose (par exemple module optionnel, ou culture générale de 40 heures) car cela ne laisse qu'une impression de langage de programmation "super-Pascal": et seul l'aspect langage serait ressenti. Avec ADA à forte dose, au contraire, plus on avance, plus l'investissement initial devient rentable : les meilleures choses se méritent et arrivent à la fin. La cohérence apparaît jour après jour aux étudiants, et il nous arrive encore de découvrir, à travers notre enseignement, des subtilités qui nous avaient échappées.

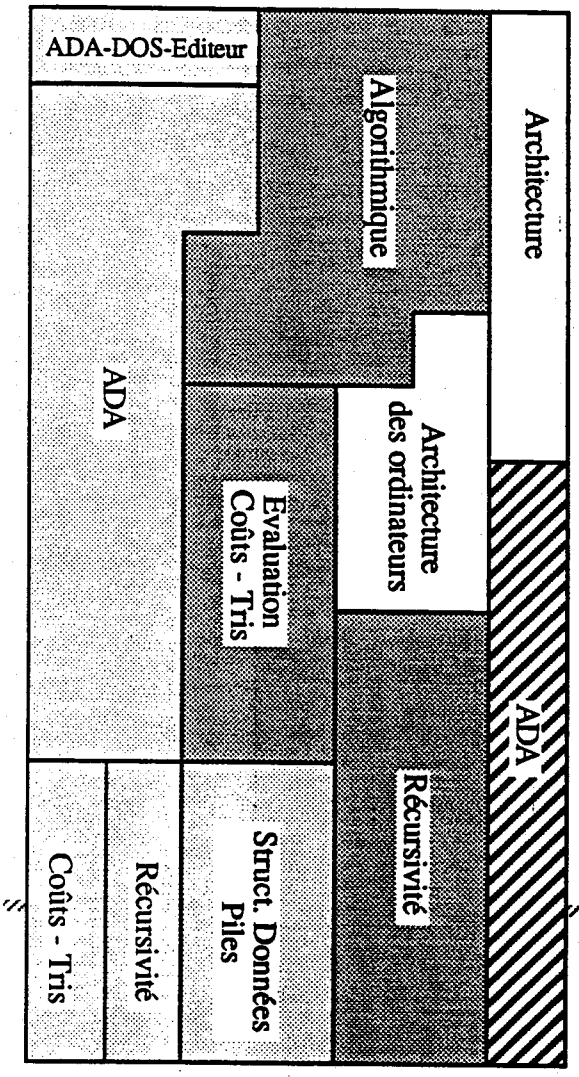
- une telle mise en oeuvre ne peut pas être le fait d'un enseignant isolé dans son département, mais doit, au contraire, être le **fruit d'un travail d'équipe.** Quant à nous, elle a concerné 6 enseignants en poste qui se sont "convertis" à ADA, auxquels il faut ajouter deux allocataires, soit plus des deux tiers de notre modeste équipe pédagogique.

XI Et la bibliographie ?

Après avoir découvert ADA grace à J. BARNES ("la référence technique"), G .BOOCH (pour l'éclairage génie logiciel) et .... "avec le sourire" (agréable pour une introduction), sans compter les divers manuels de références nous recommandons à nos étudiants l'ouvrage "Langage ADA et algorithmique" de R. OGOR et R. RANNOU chez HERMES. (cours bien structuré et rôdé à l'ENST Bretagne). Pour ses réflexions pédagogiques on n'oubliera pas le livre de M. GAUTHIER "ADA un apprentissage" chez DUNOD.

Annexe 1

Planning de l'Informatique en Première année
(Partie 1)




= 2h


Cours

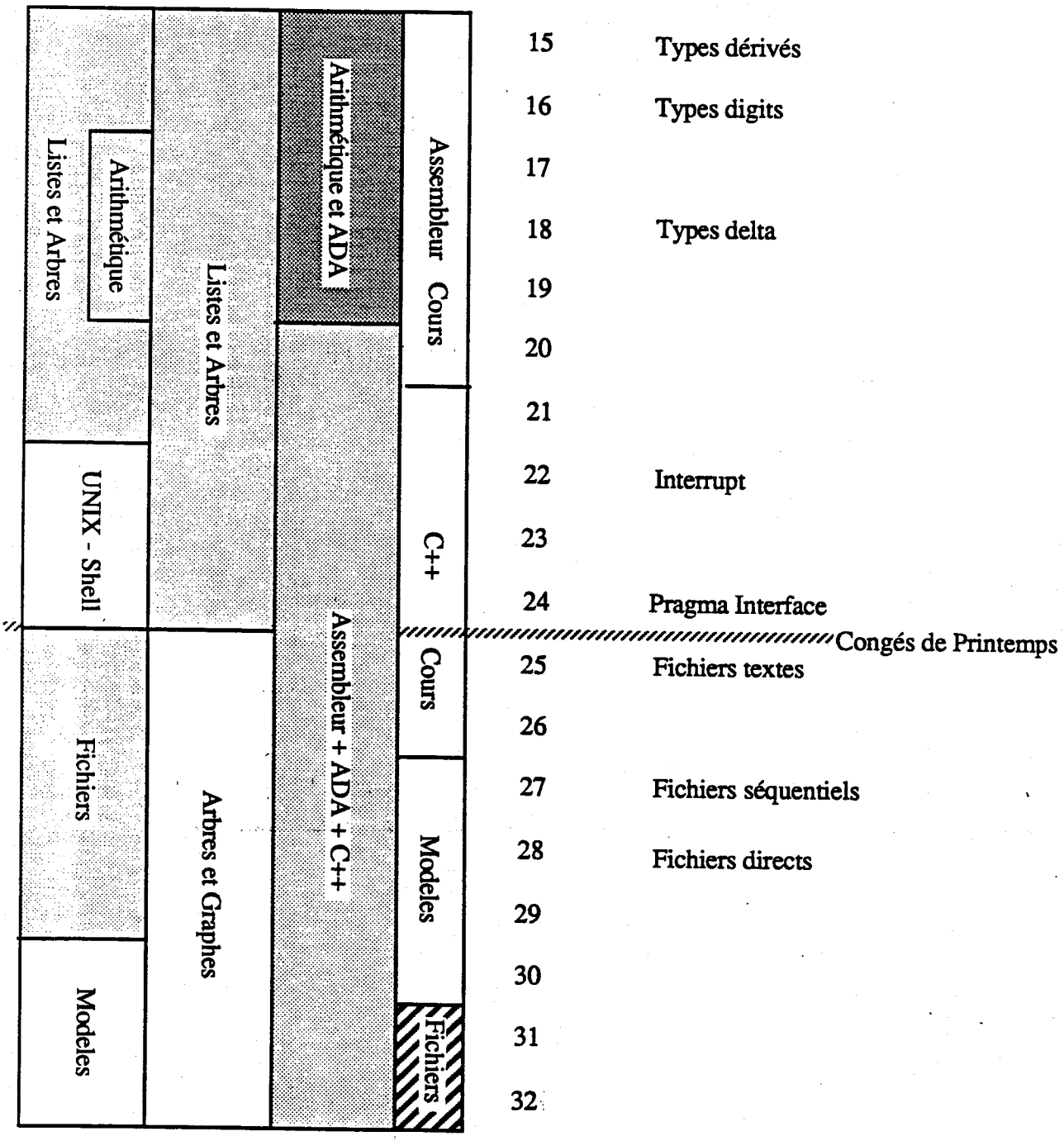

TD


TP

semaine	Notion d'ADA introduite
1	Survol : Struct. d'une unité ADA notion de bibliothèque - type prédéfini déclaration de variable - schémas alternatif et répétitif - E/S simples
2	Types scalaires - discrets - énumératifs Attributs
3	Type tableau
4	Instr. de répétition et de choix
5	Sous-programmes
6	Compilation séparée
7	Tableau non contraint
8	Type article simple
9	Paquetage
10	Type article avec discriminant
11	Paquetage et type privé
12	Généricité
	NOEL
13	Exceptions
14	Type accès

Annexe 1 (suite)

Planning de l'Informatique en Première année
(Partie 2)



L'ENSEIGNEMENT DE L'INFORMATIQUE EN PREMIER CYCLE UNIVERSITAIRE

Jean-Paul BERTRANDIAS

L'enseignement de l'informatique en Premier Cycle Universitaire
(Suite)

Jean Paul Bertrandias
Université Joseph Fourier - GRENOBLE 1

Lorsque Norbert COT m'a demandé de reprendre pour le bulletin les réflexions sur l'enseignement de l'informatique en DEUG que j'avais présentées aux Journées organisées à Lille en Septembre 1990 par Michel LUCAS, je me suis aperçu que les choses avaient beaucoup évolué en deux ans.

La première partie de l'exposé était constituée des enseignements qu'on pouvait tirer de notre expérience d'installation de l'informatique en DEUG à Grenoble. Elle s'appuyait sur le texte qui suit, rédigé avec Pierre-Claude SCHOLL et élaboré par l'équipe d'enseignants durant l'année 1989-1990.

Nous publions ce texte sans modification ; il est incomplet en ce qui concerne la deuxième année qui ne devrait pas être oubliée dans une vue complète de notre projet. D'autre part, il nécessiterait une certaine mise à jour ; la situation actuelle de réforme générale des DEUG, la reconnaissance officielle d'une mention Informatique-Mathématiques ainsi que la création d'un DEUG Technologique, ont notablement changé nos possibilités d'action et les objectifs que nous pouvons nous fixer.

La seconde partie de l'exposé était un essai de définition de l'informatique comme discipline scientifique et un développement des conséquences qu'on pouvait en tirer pour un programme d'enseignement en DEUG SSM. Je ne suis plus sûr qu'il faille présenter l'informatique uniquement comme science ..., et je préfère proposer ici une rédaction (un peu adaptée) du premier chapitre du Support d'enseignement utilisé cette année à Grenoble en première année pour essayer de donner quelques pistes significatives.

Ce premier chapitre (celui qu'on ne lit habituellement pas quand on le trouve au début d'un livre sur l'informatique !) semble cependant important sur plusieurs points :

- *bases conceptuelles* :
  - pour placer les notions fondamentales : représentations, variables, types, ...
  - pour souligner l'importance des langages,
- *représentation et codage*, avec les exemples élémentaires classiques :
  - pour familiariser avec la diversité des représentations,
  - pour les faire pratiquer sur des exercices intégrés à la culture des étudiants,
- *niveaux d'abstraction et de langage* :
  - permettant de situer et d'observer l'utilisation de divers langages,
  - montrant l'importance d'une analyse précise des problèmes.

Le parcours prévu se poursuit par :

- chapitre 2 : Information : matérialisation
- chapitre 3 ; Machines - Actions - Automates
- chapitre 4 : Langages : syntaxe, sémantique, pragmatique
- chapitre 5 : Types - Variables

- chapitre 6 : Structures de contrôle
- chapitre 7 : Structures de données
- chapitre 8 : Procédures - Fonctions

Pour la seconde partie (chapitres 6, 7, 8), Pascal est pris comme langage d'expérimentation ; la première partie permet de relativiser son importance dans la vision de l'informatique proposée aux étudiants du DEUG SSM.

Grenoble, Octobre 1992

Extrait d'une contribution à la réflexion organisée par SPECIF sur l'informatique en DEUG :
"L'informatique, discipline fondamentale en DEUG A : un exemple à Grenoble"
Avril 1990

Quelques lignes directrices sur l'installation de l'Informatique en DEUG

Jean Paul Bertrandias, Pierre-Claude Scholl
Université Joseph Fourier - GRENOBLE 1

Promotion de l'informatique en premier cycle

L'un des facteurs de l'évolution de l'informatique est l'amélioration de son image en tant que discipline fondamentale, que ce soit auprès des industriels (qui sous-estiment l'importance des aspects fondamentaux par rapport aux technologies à évolution rapide), auprès des scientifiques, universitaires notamment (qui, en général, ne perçoivent de l'informatique que ses aspects technologiques si avancés soient-ils), auprès des jeunes (mal informés du fait de la faiblesse de la formation dans le second degré) et auprès des pouvoirs publics (qui, s'ils multiplient les actions incitatives du fait des enjeux économiques et sociaux, s'obstinent à ne pas en tirer les conséquences dans les structures de décision).

Face à cette question, et pour agir sur le long terme, il nous semble essentiel de favoriser l'insertion de l'informatique fondamentale dans toutes les formations universitaires. L'introduction progressive d'enseignements d'informatique en second cycle, la mise en place de DESS Informatique Double Compétence ou de formations pour les professeurs du second degré relèvent de cette démarche, mais le premier cycle est le maillon central de cette ligne d'action.

Notre UFR s'est attaquée très activement à ce problème dans les cinq dernières années. Il a d'abord fallu obtenir un nombre d'heures suffisants dans les programmes de premier cycle, valoriser ce type de formation par rapport aux formations spécialisées de 2° et 3° cycle, organiser des équipes pédagogiques importantes (du fait du nombre d'étudiants), mais surtout inventer des contenus (en dépassant les idées standard liées à la programmation ou à l'utilisation de logiciels) et les mises en oeuvre pédagogiques appropriées.

Insertion de la discipline

Pour pouvoir enseigner l'informatique en Deug A comme discipline fondamentale, il faut d'abord assurer une insertion favorable dans l'environnement universitaire:

Un cadre administratif analogue à celui des autres disciplines
- reconnaissance par les différents conseils (du Premier Cycle, Pédagogique, Scientifique, d'Université, ...),
- mise en place d'un budget suffisant (Université, informatique pour tous, ...)
- obtention de locaux adaptés (salles de TD et salles machines proches les unes des autres, salles de maintenance, de permanence, de réunion),
- horaire global convenable (seuil qu'on peut fixer à 50 heures en 1ère année),
- coefficient suffisant (en principe, correspondant au volume horaire),
- plan concerté sur les deux années (formation générale en 1ère année, formation plus spécialisée en 2ème année dans le cadre de la réforme des Deug).

Une conformité du cadre pédagogique avec celui des autres disciplines fondamentales
- exercices progressifs et bien cadrés,
- épreuves de contrôle en temps limité, comportant des parties théoriques et pratiques, amenant à une moyenne analogue à celle des autres disciplines,
- examen de T.P. individuel.

Une concentration suffisamment forte de l'enseignement

En première année, nous répartissons les 2 heures annuelles assurées de la manière suivante: 4 heures par semaine sur une période de 12 semaines, ce qui permet deux rendez-vous différenciés par semaine: 2 heures de TD + 2 heures de programmation devant machine.

En deuxième année, un enseignement de base de 6 heures annuelles dans une section où l'informatique est une discipline majeure (Mathématiques, Physique, Informatique) et un module de 4 heures annuelles proposé dans les autres sections (5 groupes en 1990-91)

Une bonne coordination entre enseignement théorique et expérimentation

Par exemple, en première année, nous articulons l'enseignement autour de l'expérimentation pratique proposée: les notions et techniques introduites en TD sont directement illustrées par les sujets de travaux pratiques et sont immédiatement renforcées en TD après les séances pratiques.

Une reconnaissance par rapport aux autres enseignements d'informatique

Dans notre cas, l'UFR d'Informatique et Mathématiques Appliquées a donné une forte priorité à cette question, notamment en favorisant les affectations de service en premier cycle, la formation de collègues d'autres disciplines et la constitution de groupes de travail.

A propos d'objectifs

L'enseignement de l'informatique pose de multiples défis, par la variété des approches qu'elle permet en matière d'objectifs, de contenus ou de mise en oeuvre pédagogique.

Peut-on dégager des objectifs pour la pédagogie d'une discipline fondamentale, dans le cadre d'un enseignement de masse? C'est une question difficile du fait de la jeunesse de l'informatique, qu'il faut aborder progressivement au fur et à mesure des expériences. C'est dans cet esprit que nous présentons les quelques éléments suivants.

Notre hypothèse est que l'informatique peut apporter des bases culturelles utiles à l'étude des disciplines scientifiques, et que l'enseignement en DEUG A doit en développer les aspects fondamentaux, déclenchant éventuellement des vocations en direction de l'informatique.

Dans notre contexte, la cinquantaine d'heures de première année sont une sensibilisation à l'informatique dans laquelle on veut habituer les étudiants à aborder l'informatique de manière "académique" (au sens noble du terme). Ce démarrage permet d'homogénéiser les acquis des étudiants, notamment en prévision de la seconde année. Nous proposons ainsi :

- une *expérimentation* sur l'environnement informatique : matériel, système d'exploitation, langage, ...
- l'élaboration d'une *problématique* de l'informatique: formulation de problèmes-types, méthodes de réflexion et d'analyse, situations où intervient l'informatique, ...
- une réflexion sur les *modes d'expression* utilisés en informatique: problèmes, algorithmes, programmes, résultats, ...

En ce qui concerne la seconde année, nous voulons tout à la fois

- sensibiliser aux aspects fondamentaux de l'informatique au travers d'éléments théoriques, de méthodes de résolution de problèmes, et d'outils concrets,
- donner une culture générale en informatique et préparer à une compréhension des apports de l'informatique dans les disciplines scientifiques ou à une éventuelle spécialisation informatique.

Nous essayons ainsi de dégager les principes de base permettant de situer les branches assez différentes que recouvre le terme *informatique*, en évitant de tomber dans certains écueils connus:

- un enseignement purement technique (apprentissage d'un langage) ;
- un enseignement purement théorique (bases mathématiques de l'informatique) ;
- un enseignement discipline de service (utilisation de logiciels).

Au delà d'une présentation d'éléments techniques, nous pensons qu'un enseignement d'informatique doit être organisé autour des aspects méthodologiques inhérents aux problèmes de conception, d'expression et de validation. Il prépare alors aussi bien
- à une pratique expérimentale qui sera utile dans les enseignements où l'informatique est vue par ses aspects technologiques
- qu'à l'acquisition de fondements dans divers domaines: mathématiques (récurrence, algèbre, logique), langages, modélisation des informations, preuve, complexité, génie logiciel.

Un objectif global est de transmettre une attitude scientifique vis à vis des programmes, des algorithmes, des descriptions d'analyse et de problèmes,... objets susceptibles d'être manipulés par l'homme, plutôt qu'objets d'exécution par une machine. La motivation première des étudiants est le plus souvent liée à la machine. Il faut la réorienter vers l'activité humaine. Il faut faire comprendre aux étudiants le besoin d'une discipline. Il faut équilibrer leurs goûts et leurs énergies entre les problèmes de conception, la réutilisation adéquate de techniques répertoriées et la mise en oeuvre concrète.

Un élément essentiel dans la structuration de l'apprentissage est une claire perception des divers niveaux de complexité qui caractérisent l'informatique : complexité factuelle, inhérente aux problèmes et aux limites des machines, complexité technique liée à la variété des outils et à leur sophistication, et complexité conceptuelle, imposée par la distance entre l'énoncé d'un problème et une solution informatique. Il faut repérer ces divers types de complexité et placer des lignes directrices simples permettant de leur faire face: classification des types de problèmes, organisation méthodique et hiérarchique des connaissances techniques, mise en évidence de méthodes d'analyse, approche pratique expérimentale.

Les multiples savoir-faire informatiques s'appuient sur la maîtrise des modes d'expression employés, aux divers niveaux d'abstraction considérés, et sur des attitudes rigoureuses et précises, même si elles ne sont pas formelles. On trouve là une difficulté majeure rencontrée par les étudiants qui ont une tendance naturelle à ne s'intéresser au problème d'expression qu'au moment de rédiger un programme dans un langage de programmation particulier (voire devant la machine!). Nous pensons qu'un cours d'informatique doit mettre un accent important sur la notion même de langage, en insistant sur les aspects pragmatiques, au plan conceptuel pour l'expression de la pensée, et au plan technique pour la mise en oeuvre d'une solution.

A propos de mise en oeuvre

La mise en oeuvre pédagogique d'un enseignement d'informatique est affaire délicate notamment en matière d'équilibre entre aspects fondamentaux et pratiques, et de contrôle des activités proposées dans des directions nécessairement multiples: résolution de problèmes, rigueur d'expression, maîtrise des techniques, pratique des machines au travers de la programmation, ...

Pour animer une progression pédagogique qui intègre les divers objectifs ci-dessus, il faut éviter l'énumération fastidieuse des techniques, savoir choisir parmi la diversité des sujets à aborder et renouveler les thèmes d'application. Enfin, il faut d'année en année intégrer les progrès conceptuels et technologiques de l'informatique. C'est ici qu'intervient la notion d'équipe. Les résultats sont d'autant plus riches si les participants peuvent apporter les contributions de domaines variés de l'informatique.

Dans ce qui suit, nous proposons quelques réflexions sur la base de l'expérience menée en **première année** avec 5 sections de 4 groupes (environ 720 étudiants) (*).

(*) 6 sections de 4 à 5 groupes en 1991-92

Unité pédagogique

Le nombre d'étudiants intervient sur le nombre de groupes à gérer, le nombre d'enseignants concernés, la diffusion des documents s'ils sont communs et le retentissement de tout problème...

Ce nombre fait ressortir fortement la question de l'unité pédagogique: comment mettre en oeuvre une politique commune à tous les étudiants du Deug A ? Faut-il unifier les groupes ? les sections ? Quel degré de proximité adopte-t-on entre les groupes ?

Si l'on choisit d'organiser l'enseignement par section, par exemple sur la base d'un programme commun, on réduit la lourdeur de la coordination, mais au détriment d'un consensus réel sur l'enseignement. Il devient difficile de comparer les acquis et de faire évoluer les enseignements de manière cohérente d'année en année. On risque de propager ainsi une hétérogénéité de connaissances inévitable du fait de l'indépendance des sections. On peut enfin estimer que l'investissement pédagogique est finalement plus important, par la duplication de certaines taches et la difficulté de mise en commun de matériel pédagogique.

Nous avons choisi d'une part d'organiser l'enseignement uniquement sur la base de TD et de TP (pas de cours magistral) et d'autre part de coordonner fortement les 20 groupes concernés. Ceci nécessite un grand consensus de l'équipe pédagogique sur les objectifs et les méthodes d'enseignement. Il est notamment fondé sur des documents de référence, pour la coordination entre les enseignants ou destinés aux étudiants. Ces documents doivent concerner la matière, mais aussi chacune des notions ou techniques présentées, la progression des TD et leur conduite, le matériel et son utilisation ...

Dans notre contexte expérimental, ces choix ont permis une meilleure appréciation de la démarche envisagée en favorisant une concertation sur la nature des exemples à présenter et en permettant une synthèse immédiate après les expériences de chacun. Cette mise en oeuvre commune évite les déviations produites par la division en plusieurs sections et facilite très sensiblement l'intégration d'enseignants nouveaux et notamment des jeunes (moniteurs).

Le principal écueil réside dans la lourdeur de coordination et de gestion. Les groupes doivent progresser en parallèle avec des points d'appui soigneusement choisis. Malgré cela, les écarts pédagogiques persistent mais peuvent être discutés explicitement.

On doit s'appuyer sur une structure de coordination aussi légère et efficace que possible. On est ainsi conduit à une organisation hiérarchisée car on ne peut coordonner 20 à 25 personnes à la fois en direct. Il faut se donner les moyens de transmettre d'une année à l'autre l'investissement ainsi réalisé, par exemple en définissant un cahier de compte rendu chronologique, une batterie d'exercices ciblés et commentés, une mémoire d'exemples et de sujets, une bibliothèque de référence,..., le tout étant enrichi progressivement. L'investissement pédagogique correspondant, le travail matériel important de rédaction et de relecture peuvent être répartis sur l'équipe.

Il faut assurer une correction adéquate des TP. On peut s'interroger sur l'opportunité de "corrigés types": ils sont toujours réducteurs de la réalité, car ils ne donnent qu'une vision partielle des multiples approches possibles d'un problème et ne peuvent, pour des raisons matérielles, expliciter suffisamment les raisonnements qui sous-tendent l'élaboration de la (ou des quelques) solution(s) présentée(s).

On peut déjà percevoir que ces conditions de travail peuvent difficilement être considérées comme une norme car il n'est pas dans la tradition de nos universités de définir avec une telle précision les objectifs et les méthodes pédagogiques qu'un groupe s'impose et impose aux différents enseignants, même si le consensus est souvent recherché.

L'hétérogénéité des acquis en informatique

Quelques étudiants (1 à 2%) ont suivi une option informatique au lycée. Cette préformation ne développe pas automatiquement un désintérêt. On pourrait imaginer un aménagement de leur travail. Les autodidactes du BASIC (plus de 20% des étudiants) peuvent introduire le trouble ou au contraire être à l'origine de bonnes questions. L'organisation que nous pratiquons permet de les intégrer rapidement à notre progression pédagogique, de leur faire percevoir la nouveauté de la démarche, et même de récupérer certaines de leurs compétences au profit de tous.

Extrait du **Support de Travaux Dirigés d'Informatique - DEUG A 1ère année**
Université Joseph Fourier - GRENOBLE 1
Septembre 1992

Françoise et Jean Paul Bertrandias
Université Joseph Fourier - GRENOBLE 1

INFORMATION

REPRESENTATION - TRAITEMENT

L'*informatique* est en rapport avec l'*information.*

Ce premier chapitre donne d'abord quelques points de repère sur la notion difficile d'*information.* Elle est relative à la connaissance et à la manière dont on peut élaborer et transmettre cette connaissance. Cela se fait par l'intermédiaire de *langages* dont on verra apparaître progressivement l'importance - à un point tel que l'on pourrait dire que l'***informatique** a principalement pour objet les **langages** dans leurs fonctions de représentation et de matérialisation.*

1 - Information

Une information est une *connaissance explicitée et partagée* sur une réalité qu'on a bien délimitée. La notion d'information se place d'une part au *niveau de la connaissance* qu'on peut avoir de la réalité et d'autre part au *niveau du langage* qui permet d'expliciter et de partager cette connaissance.

a - Niveau de la connaissance

Pour rendre compte de la réalité, la pensée élabore idées ou notions ou *concepts* par l'opération d'*abstraction.* La pensée saisit ce qui fait la singularité et l'unité d'une chose, d'un ensemble de choses, d'une relation, ..., et cela constitue un concept défini en *compréhension*, ou encore elle définit un concept en saisissant ce qui est commun à plusieurs choses ou concepts, qui forment l'*extension* du concept ; elle peut aussi procéder par *construction* en composant des concepts déjà connus.

Dans cette élaboration, les relations qui sont saisies intellectuellement par les concepts "donnent une idée" des relations pouvant exister effectivement entre choses ou états de choses ou événements et en sont ainsi une *représentation abstraite.*

Une réalité qu'on a bien isolée et délimitée par un concept peut apparaître sous différents aspects (appartenant à l'extension du concept) et ces aspects sont eux-mêmes précisés par des concepts par exemple *quantitatifs* (valeurs entières, réelles, ...) ou *qualitatifs* (couleurs, catégories, ...). Une *information* sur cette réalité est la liaison qui est établie (au niveau des concepts) entre la réalité elle-même et l'aspect sous lequel elle apparaît effectivement.

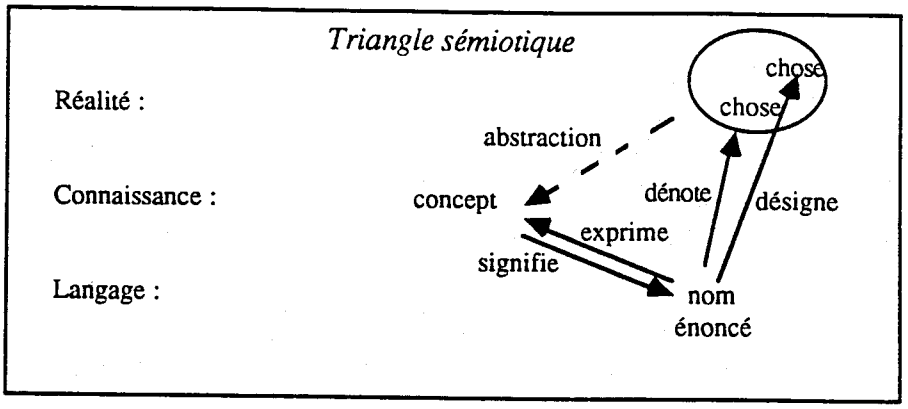
Dans un contexte scientifique, les concepts quantitatifs ou qualitatifs s'appellent des *valeurs*, le concept considéré s'appelle une *variable* et l'ensemble des valeurs susceptibles d'apparaître s'appelle le *type* de la variable. Ce qui constitue l'information relative à une variable est donc l'association qui est établie, conformément à ce qui existe effectivement, entre cette variable et une valeur particulière (qui est "variable" dans le type).

A ce niveau, le type est un *ensemble abstrait* qui peut être défini en extension mais qui est souvent un ensemble mathématique (voir chapitre V : Types - Variables). Des relations (d'équivalence, d'ordre, ...) et des opérations (arithmétiques, logiques, ...) sont définies à partir de la réalité (c'est-à-dire d'abord indépendamment des "mathématiques") sur le type ; elles permettent *le traitement des valeurs* : calculs, classements, regroupements, ...

b - Niveau du langage

Une information ne peut s'expliciter et se transmettre que dans un *langage* (oral, écrit, gestuel, ...) et donc par l'intermédiaire de *symboles* matérialisés (paroles, mots, gestes, ...).

Dans le cadre d'un langage, un concept est *exprimé* par un *nom* (ou plus généralement un *énoncé*). Le concept donne la *signification* (ou le sens), de ce nom. Le nom *exprime* (la signification du) le concept, *dénote* l'ensemble des réalités visées et *désigne* chacune d'entre elles. Le nom est une *représentation* de cette réalité, reflet dans le langage de la représentation abstraite donnée par le concept.



Dans un langage spécialisé, l'information relative à une variable s'exprime donc par une liaison entre le nom de la variable et le nom de la valeur qui lui est associée actuellement. C'est sous cette forme que l'information intervient normalement en informatique (voir chapitre V : Types - Variables). Les *significations* (celle du nom de la variable et celle du nom de la valeur) sont en apparence éliminées. Mais elles ne doivent pas être perdues de vue dans les applications !

Une caractéristique de l'information est le fait que les noms (au moins ceux des valeurs) doivent être *matérialisés*, d'abord pour l'*explicitation* et le *partage* entre interlocuteurs dans un langage. C'est cette matérialisation qui permet la *transmission* des valeurs (par le son, l'image, des signaux acoustiques ou électriques, ...). Elle permet aussi leur *mémorisation* : la valeur peut rester disponible par son nom dans un endroit dans un endroit matériel auquel on doit avoir *accès* pour retrouver l'information complète (la variable et les significations étant supposées non perdues ..).

Dans ce chapitre, on s'intéresse à la *représentation* et au *traitement* des valeurs ; on laissera la *matérialisation* pour le chapitre II.

2 - Représentation des valeurs

Les valeurs sont exprimées chacune par des *noms* dans le cadre d'un *langage* précisé (langue naturelle parlée, langue naturelle écrite, mathématiques, langage de programmation, ...). Le nom est une *représentation* de la valeur et sa forme donne en général une idée de ce qu'est la valeur, ou de la manière dont elle se construit à partir de valeurs plus simples.

Exemple : la valeur entière nommée "vingt-et-un" dans le langage courant se nomme

XXI chez les Romains, ▯▯▯▯▯ dans les bureaux de vote,

21 ou $(25)_8$ ou $(15)_{16}$ ou $(10101)_2$ en mathématiques, ...

En informatique, la *désignation* des valeurs se fait par un *codage* qui est une représentation particulière destinée à faciliter les *matérialisations* et à donner une grande efficacité aux *traitements*.

a - Valeurs de vérité - Bits

L'information "la plus petite possible" s'exprime par une *proposition* sur une réalité, la valeur qu'on attend alors étant l'une des deux *valeurs de vérité* : soit *vrai* soit *faux* (ou, de manière équivalente, une *question* dont la réponse peut être soit *oui* soit *non*). On s'est mis d'accord sur la signification de la proposition, c'est-à-dire sur la représentation (le "tableau", l'"image") de la réalité donnée par la proposition. L'information sur la réalité visée actuellement est le fait que cette

représentation est conforme (la proposition a la valeur "vrai") ou non conforme (la proposition a la valeur "faux") à ce qui existe effectivement.

Les deux valeurs de vérité se représentent dans le langage courant par leurs noms (vrai/faux, oui/non, true/false, ...) ou par des abréviations (V/F, O/N, T/F, Y/N, ...).

En informatique, les deux valeurs de vérité s'appellent les *valeurs booléennes* et se codent par un chiffre binaire 1 ou 0, qu'on appelle un *bit* (pour : *bin*ary dig*it*). Leur ensemble s'appelle le *type booléen* et des opérations sont définies naturellement sur ce type (voir chapitre III : Machines - Actions - Automates).

b - Représentation à l'aide de valeurs de vérité

Dans le cas d'informations plus "riches", l'ensemble des valeurs possibles est plus grand et doit être bien défini ; on peut *nommer* chaque valeur et le type est alors simplement déterminé par l'*énumération* des noms (type fini - exemple : caractères) ou par la manière dont ils se *construisent* (type éventuellement infini - exemple : entiers).

Pour la représentation des valeurs, il faudrait donc examiner la manière dont les noms sont *formés* en tenant compte de la *signification* des valeurs (voir chapitre IV : Langages). Dans ce chapitre, on se contentera de généralités sur le codage en informatique avec l'application aux deux types les plus usuels (§ c et d).

Pour le codage des valeurs, on peut toujours se ramener à un codage par des bits :

toute valeur peut s'exprimer au moyen de valeurs de vérité,

ce qui provient du principe plus philosophique :

toute réalité exprimable et vérifiable peut être déterminée par des propositions.

Dans le cas où le type est un ensemble fini, cela consiste à procéder de la façon suivante :

on définit une subdivision de l'ensemble en deux parties qui sont elles-mêmes divisées chacune en deux parties qui sont elles-mêmes ... ; les propositions ou questions portent sur l'appartenance à l'une ou l'autre des parties de chaque division.

Exemple : l'information est l'âge de tel étudiant
(en nombre entier d'années entre 0 et 100)

Le type est l'intervalle d'entiers [0, 100] qu'on divise au moyen des puissances de 2.
Dans le cas où l'âge a pour valeur 21, on obtient :

Age ≥ ...	64	32	16	24	20	22	21
Valeur	faux	faux	vrai	faux	vrai	faux	vrai

On peut vérifier facilement que, si l'ensemble a N éléments et si k est l'entier tel que $2^{k-1} < N \leq 2^k$, il suffit de k questions pour déterminer un élément (en divisant l'ensemble en parties assez équilibrées) et qu'inversement il n'est pas possible de déterminer un élément dans tous les cas en moins de k questions.

Par un procédé de ce genre, toute valeur est représentée par une suite de valeurs de vérité, ou, avec le codage par les chiffres, par une suite de bits.

Dans l'exemple, k est égal à 7 ; l'âge 21 est représenté par la suite faux, faux, vrai, faux, vrai, faux, vrai et peut se coder par la suite de bits 0010101.

L'entier k (égal à la valeur entière par excès de $\log_2 N$) chiffre la *quantité d'information* nécessaire pour définir une valeur de ce type. C'est le nombre minimum de propositions qu'il faut utiliser pour déterminer la valeur de l'information (cf. Exercice 7 : Transmission d'octets).

c - Codage des caractères - Octets

Le langage écrit habituel utilise un certain nombre de *symboles* élémentaires :
- 26 lettres majuscules : A, ... , Z,
- 26 lettres minuscules : a, ... , z, ainsi que des lettres accentuées à, é, è, ...
- 10 caractères chiffres : 0, ... , 9,
- une trentaine d'autres caractères utilisés dans l'écriture (ponctuation, signes d'opérations, ...),

soit une centaine de caractères ; puisque $100 > 2^6$, il faut donc au minimum 7 bits pour le codage.

Dans la pratique, l'ensemble des caractères est codé sur *8 bits*, le bit de plus fort poids (bit de gauche) étant égal à 0 (dans certains cas, ce bit peut être utilisé pour des tests de contrôle de transmission).

--> Exercice 6 : Détection et correction des erreurs de transmission (*)
--> Exercice 7 : Transmission d'octets

Le codage des caractères sur 8 bits explique le fait que l'unité de codage utilisée dans l'organisation matérielle des ordinateurs actuels (voir chapitre II) est l'*octet* qui est un groupement de 8 bits.

Il existe plusieurs codages classiques des caractères. Plutôt que par des propositions (c'est une lettre, c'est une minuscule, c'est un chiffre, ...), ils sont construits en associant à chaque caractère un entier entre 0 et 127 qui est son code, cela de façon à retrouver facilement certaines propriétés des caractères à partir de leur code (ordre alphabétique, correspondance entre un chiffre décimal et l'entier qu'il désigne, différence entre majuscule et minuscule, ...). En fin de chapitre, on trouvera le codage le plus courant (**Annexe 1** (*) : Codage ASCII).

d - Codage des entiers

Dans l'exemple du paragraphe 2.b, la valeur entière 21 est représentée par la suite de bits 0010101. On reconnaît l'*écriture de l'entier* 21 *en base* 2, ce qui s'explique facilement quand on examine la manière dont la suite a été obtenue. C'est ce codage qui est utilisé pratiquement.

En informatique, les **entiers positifs** sont codés en général par leur écriture en base 2.
On trouve en fin de chapitre un rappel sur le développement des entiers dans une base quelconque (**Annexe 2**).
D'autres codages des entiers sont utilisés pour certaines applications (cf. Exercice 3 : code de Gray).
--> Exercice 1 : Longueur du codage d'un entier positif

Dans la pratique, les entiers sont codés sur un nombre fixé d'octets, souvent sur 4 octets. Pour certaines applications, le nombre d'octets peut être arbitrairement grand (mais dans les limites imposées par le matériel : calcul en précision "infinie" !).

Il est courant de noter les entiers dans le système *hexadécimal* (base 16) dont les chiffres sont
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f .
Chaque chiffre hexadécimal correspond à 4 bits et chaque octet de la représentation d'un entier correspond à deux chiffres de son développement hexadécimal, ce qui est plus commode que 8 chiffres binaires !

Exemple : L'entier 77 a pour décompositions : $77 = 2^6 + 2^3 + 2^2 + 2^0$
$= 4*16^1 + 13*16^0$.
Il est donc représenté par 100 1101 en base 2 et 0100 1101 sur un octet,
4 d en base 16.

--> Exercice 2 : Ecriture des entiers en base 16

Pour le codage des **entiers de signe quelconque**, il existe plusieurs règles ; la plus utilisée (dite règle du *complément* à 2) est étudiée en exercice.
--> Exercice 4 : Codage des entiers négatifs
--> Exercice 5 : Opérations sur les entiers de signe quelconque

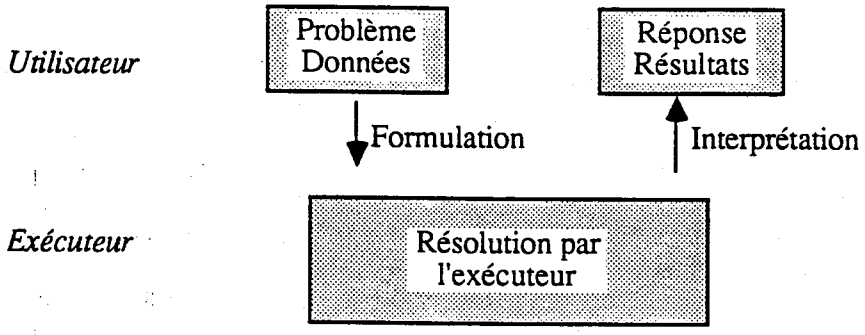
(*) Les exercices et les annexes ne sont pas reproduits mais sont disponibles !

3 - Traitement des valeurs

a - Résolution de problème

Les valeurs sont destinées à être *traitées* : on doit faire des calculs, effectuer des classements, regrouper des valeurs en valeurs plus élaborées, ... Ces opérations sont réalisées en principe par un *exécuteur* (en général une *machine)* qui effectue l'*action* de traitement.

Ce processus intervient dans un contexte de *résolution de problème*. Un *problème* comporte des valeurs comme *données* et pose des questions à leur sujet ; le traitement consiste à leur faire subir un certain nombre de transformations pour élaborer de nouvelles valeurs qui sont les *résultats* du traitement et donnent une *réponse* au problème.



La formulation du problème doit évidemment tenir compte de ce que sait faire l'exécuteur :

Exemple : Calcul d'une racine carrée

Dans les questions de géométrie pratique interviennent des distances, donc des racines carrées.
Le problème auquel on s'intéresse est le calcul des racines carrées ; l'exécuteur est une machine obéissant à un langage de programmation.
La plupart de ces langages de programmation fournissent une fonction "racine carrée" ou même des moyens de niveau plus élevé.

Exemple : langages de type fonctionnel (Miranda, ...)
Le langage permet de résoudre certaines équations.

```
... x, a = réels ;
    lire (a) ;
    x * x  = a ; x > 0 ; (* écriture de la définition de √a *)
    écrire(x) ; ...
```

Si le langage ne fournit pas la fonction "racine carrée", on doit donner explicitement un procédé de calcul : règle arithmétique, suite récurrente, ...

Exemple : langage de type actionnel (Basic, Pascal, ...)

```
... x, a : réels ;
   lire(a) ;
   x:= a ;              (* initialisation *)
   faire 6 fois
              x:= (x + a/x)/2 ;
   écrire (x) ; ...
```

Dans la pratique, la phase de *formulation* consiste souvent à *traduire* le problème en un problème acceptable par l'exécuteur et celui-ci doit alors le *reformuler* pour le confier à un exécuteur de *niveau* inférieur, etc ..., les réponses aux différents niveaux étant alors *interprétées* successivement jusqu'à donner une réponse au problème initial. On reviendra sur ce point dans le chapitre sur les langages (chapitre IV) car les démarches de *formulation* et d'*interprétation* sont souvent en fait des démarches de *traduction* d'un langage dans un autre.

Cette décomposition en niveaux est *extrêmement importante* aussi bien pour l'utilisation de l'informatique que pour la conception de systèmes informatiques.

En effet, lors de la formulation d'un problème en vue de sa résolution par un exécuteur (ou une machine), il faut arriver à savoir *quoi faire* en connaissant ce que *sait faire* l'exécuteur et en lui laissant le soin du *comment faire*. Ce qu'il suffit de connaitre sur l'exécuteur est donc sa *spécification*, c'est-à-dire une description de ce qu'il peut exécuter ; on n'a pas à entrer dans la manière dont il le fait. On fait *abstraction* des détails de l'exécution elle-même et de la façon dont l'exécuteur "tient" la spécification ; on peut se concentrer sur des aspects plus essentiels et diminuer ainsi la *complexité* du problème.

Dans un contexte de conception de systèmes de traitement de l'information, la notion d'exécuteur a l'avantage de circonscrire le travail dans des limites bien précises : celui qui conçoit un système ou une partie de système doit en réaliser exactement la spécification en tenant compte des outils plus élémentaires (et bien spécifiés !) qui sont mis à sa disposition.

On voit ainsi apparaître un schéma en *niveaux emboîtés* qui peuvent s'interpréter de façon diverse :
- niveaux d'abstraction en ce qui concerne les aspects théoriques,
- niveaux de complexité en ce qui concerne les aspects pratiques,
- niveaux de technologie dans une mise en oeuvre matérielle,
- niveaux d'expression artistique, esthétique, ...

--> **Exercice 9** : Comptabilité à diverses échelles
--> **Exercice 10** : Orgue de Barbarie

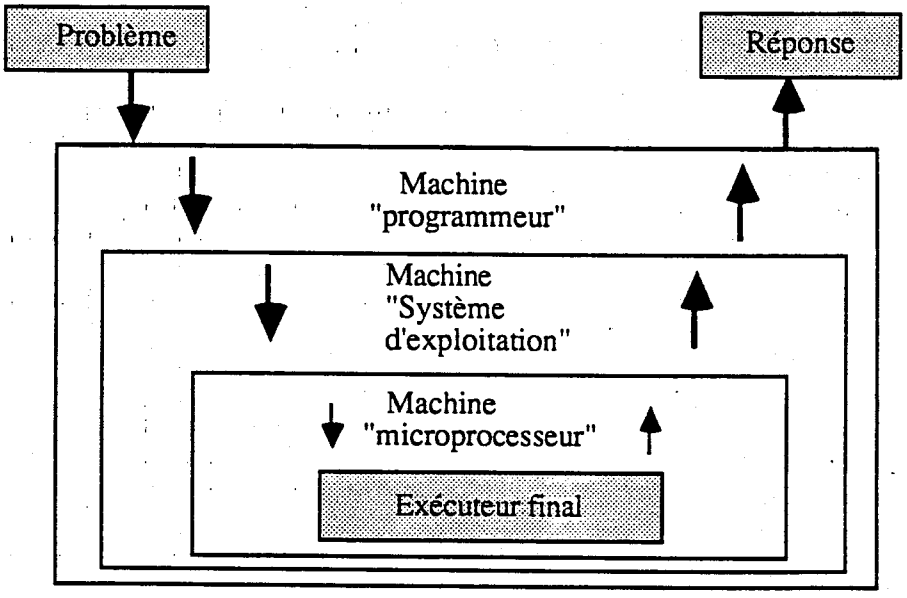
b - Niveaux d'utilisation des ordinateurs

Dans le cadre de l'informatique, l'exécuteur privilégié est l'*ordinateur* qui, assisté de ses périphériques, assure les fonctions de traitement. Mais son utilisation d'une part, son fonctionnement d'autre part font apparaitre beaucoup de niveaux intermédiaires. A titre d'exemple, on peut en détailler quelques uns dans l'utilisation qu'on en fera dans l'enseignement de DEUG A 1ére année : résolution de problèmes en utilisant Pascal par exemple comme langage de programmation.

Ces niveaux mettent en évidence trois *exécuteurs* ou "machines" :

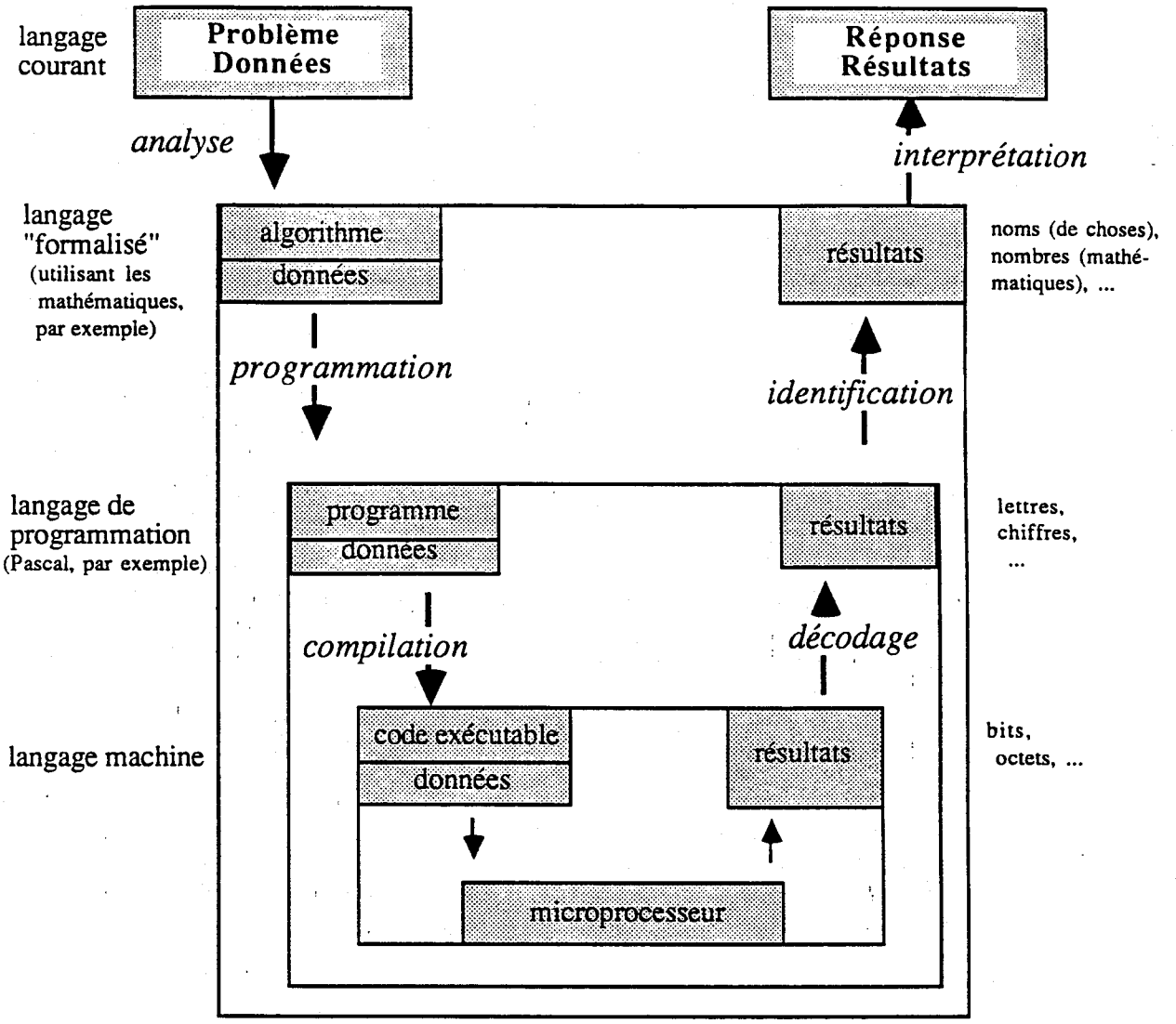
- la machine "programmeur":
  le programmeur et son ordinateur,
- la machine "système d'exploitation" :
  l'ordinateur muni de son système d'exploitation,
- la machine "microprocesseur" et ses périphériques.

Cela peut se visualiser par le schéma :



Ces niveaux peuvent aussi se caractériser par les divers *langages* dans lesquels le problème est formulé : langage courant, langage formalisé, langage de programmation, langage machine.

Le schéma peut alors se détailler de la façon suivante :



Dans le paragraphe suivant, on va "donner à voir", sur un problème concret, les différentes formulations intervenant dans ce schéma.

Remarque

On peut sur ce schéma placer différents domaines touchant à l'informatique. En partant du bas :

- Physique du solide,
- Conception des circuits élémentaires - VLSI,
- Conception de systèmes - Systèmes temps réel,
- Administration de systèmes et de réseaux,
- Conception de langages de programmation, de compilateurs, ...
- Programmation en langage évolué,
- Génie logiciel,
- Analyse, informatique de gestion,
- Sociétés de service en informatique,
- ...

4 - Exemple de résolution de problème : Calcul de la moyenne olympique

Langage courant : énoncé du problème

Dans certaines épreuves sportives, 8 juges donnent pour chaque athlète une note sur 100. La note finale ("moyenne olympique") attribuée à l'athlète est obtenue en éliminant la plus forte et la plus faible des 8 notes et en faisant la moyenne (entière) des 6 autres.

Calculer la moyenne olympique à partir des 8 notes données successivement.

Analyse : proposition d'une méthode de résolution

- Identification des informations (= variables) pertinentes :
  une information Note prenant successivement 8 valeurs, ou 8 informations Note1, Note2, ..., Note8 ? ...
- Indication des traitements à réaliser :
  Analyse fonctionnelle : une fonction de 8 variables à définir :
  quelles fonctions intermédiaires introduire (somme, maximum, minimum, ...) ?
  Analyse actionnelle : quelle action répétitive proposer :
  acquisition de toutes les notes puis traitement global, ou 8 mises à jour après acquisition ? ...

Langage formalisé : algorithme (actionnel)

Variables de type entier : Note : note venant d'être acquise
S : somme des notes déjà acquises
Max : plus forte des notes déjà acquises
Min : plus faible des notes déjà acquises

Initialisation de S à 0, de Max à 0 et de Min à 100.

Faire 8 fois : Acquisition de Note ;
Mise à jour de S ;
Mise à jour éventuelle de Max ;
Mise à jour éventuelle de Min .

Calcul de (S - Max - Min) divisé par 6.

Programmation : traduction dans le langage Pascal

- Choix sur l'acquisition des données et la présentation du résultat, ...
- Choix de la forme de la répétition et des mises à jour, ...
- Recherche de l'efficacité (nombre de comparaisons, ...).

Langage de programmation :

programme en Pascal

```
program MOYOLYMP ;
var S, Max, Min, Note: integer;
    I: integer;
begin
    S:=0 ; Max:=0 ; Min:=100 ;
    for I:=1 to 8 do
    begin
          readln(Note) ;
          S:=S+Note ;
          if Note>Max then Max:= Note;
          if Note<Min then Min:= Note;
    end;
    writeln((S-Max-Min) div 6)
end.
```

programme en C

```
main()
{static int S, Max, Min, Note;
 register int I;

 S=0; Max=0; Min=100;
 for (I=1 ; I<=8 ; I++)
    {
     scanf("%d" , &Note);
     S=S+Note;
     if (Note>Max) Max=Note;
     if (Note<Min) Min=Note;
    }
 printf("%d",(S-Max-Min)/6);
}
```

Compilation : traduction en code exécutable sur microprocesseur

- Appel du *compilateur* par l'intermédiaire du système d'exploitation ("compile")
  --> traduction du programme en *langage d'assemblage* ou *langage machine* ;
  production du *code binaire* : les lecture/écriture sont laissées en "blanc".
- Appel de l'*éditeur de liens* ("link")
  --> production du *code exécutable* en langage machine : les programmes de lecture/écriture sont *liés* au programme pour le rendre exécutable par le microprocesseur et les translations d'adresse sont faites (dans le cas d'une exécution à une adresse fixe) - voir chapitre II.

Langage machine :

Forme du programme après la compilation et avant l'édition de liens (code binaire) :

Adresses	*Octets*																
00000000	07	01	10	00	0a	00	30	00	00	00	aa	00	00	00	00	00	
00000010	00	00	00	00	00	00	00	00	00	00	06	00	00	00	08	00	EN-TETE
00000020	00	00	00	00	00	00	42	00	00	00	72	00	00	00	00	00	
00000030	4e	56	ff	fc	2f	07	42	79	00	00	00	b0	42	79	00	00	
00000040	00	b2	70	64	33	c0	00	00	00	b4	7e	01	0c	47	00	08	
00000050	6e	5a	48	79	00	00	00	b6	48	79	00	00	00	aa	4e	b9	
00000060	00	00	00	00	50	4f	30	39	00	00	00	b0	d0	79	00	00	
00000070	00	b6	33	c0	00	00	00	b0	30	39	00	00	00	b6	b0	79	TEXTE
00000080	00	00	00	b2	6f	0a	33	f9	00	00	00	b6	00	00	00	b2	
00000090	30	39	00	00	00	b6	b0	79	00	00	00	b4	6c	0a	33	f9	
000000a0	00	00	00	b6	00	00	00	b4	52	47	60	a0	30	39	00	00	
000000b0	00	b0	90	79	00	00	00	b2	90	79	00	00	00	b4	48	c0	
000000c0	81	fc	00	06	3f	00	48	79	00	00	00	ad	4e	b9	00	00	
000000d0	00	00	5c	4f	2e	1f	4e	5e	4e	75	25	64	00	25	64	00	
000000e0	6d	61	69	6e	5f	00	00	00	00	00	00	00	00	00	00	00	
000000f0	10	00	00	00	00	00	73	63	61	6e	66	5f	00	00	00	00	CHAINES
00000100	00	00	00	00	00	00	1a	00	00	00	00	00	70	72	69	6e	
00000110	74	66	5f	00	00	00	00	00	00	00	00	00	1a	00	00	00	
00000120	00	00	45	00	00	08	00	45	00	00	0e	00	45	00	00	16	
00000130	00	45	00	00	24	00	44	00	00	2a	00	47	00	00	30	00	
00000140	01	00	45	00	00	38	00	45	00	00	3e	00	45	00	00	44	RELO-
00000150	00	45	00	00	4a	00	45	00	00	50	00	45	00	00	58	00	CATION
00000160	45	00	00	5c	00	45	00	00	62	00	45	00	00	68	00	45	
00000170	00	00	70	00	45	00	00	74	00	45	00	00	7e	00	45	00	
00000180	00	84	00	45	00	00	8a	00	44	00	00	98	00	47	00	00	
00000190	9e	00	02	00													

Partie correspondant au programme C :

Adresses	*langage machine*	*langage d'assemblage*	*Commentaires*
L0 =0	4e 56 ff fc	link a6, $-4	Lien entre le système et le programme
	2f 07	move.l d7, -(a7)	Sauvegarde du contenu du registre d7
	42 79 00 00 00 b0	clr L2	Mise à 0 de S
	42 79 00 00 00 b2	clr L3	idem pour Max
	70 64	moveq $100, d0	Transfert de 100 dans le registre d0
	33 c0 00 00 00 b4	move d0, L4	puis à la place réservée pour Min
	7e 01	moveq $1, d7	Initialisation à 1 de l'indice I
L8 =1c	0c 00 08	cmpi $8, d7	Comparaison de I à 8 = début de la boucle
	6e 5a	bgt.s L6	Si I > 8, saut après la fin de la boucle
	.....		
	4e b9 00 00 00 00	jsr scanf_	Acquisition de la valeur de Note
	.....		(adresse en "blanc")
	52 47	addq $1, d7	Incrémentation de l'indice I
	60 a0	bra.s L8	Branchement au début de la boucle
L6 =7c	30 39 00 00 00 b0	move L2, d0	...
	.....		...
	2e 1f	move.l (a7)+, d7	Restitution du registre d7
	4e 5e	unlk a6	Suppression du lien avec le système
	4e 75	rts	Fin du programme
	.....		
L2 =b0			Place réservée pour la valeur de S
L3 =b2			Place réservée pour la valeur de Max
L4 =b4			Place réservée pour la valeur de Min
L5 =b6			Place réservée pour la valeur de Note
			Le registre d7 est réservé pour la valeur de l'indice I

Remarque : Dans cet exemple, les étapes et les textes ont été simplifiés pour des raisons pédagogiques. Les choix qui ont été faits ne sont pas des modèles et pourraient être améliorés par la suite : les types ne sont pas discutés, le traitement itératif pourrait se faire de manière plus systématique et plus efficace, ...

CONTRIBUTIONS A L'INFORMATIQUE

Cette rubrique contient des contributions à l'art ou à la science informatique. Bien entendu, celles-ci n'engagent que leurs auteurs. Le Bulletin Spécif est ouvert aux points de vue différents ou aux remarques qu'elles pourraient susciter.

Bulletin Spécif

Analyse de quelques contributions de la Théorie de la Programmation

Guy Cousineau
LIENS
45 rue d'Ulm
75005 Paris
cousineau@dmi.ens.fr

Ce qui suit est le texte d'une conférence donnée dans le cadre de la "Journée en l'honneur de Jacques Arsac" le 23 octobre 91 à la Sorbonne. Son ambition était volontairement limitée. Il ne s'agissait en aucune façon dresser un bilan exhaustif des contributions de la Théorie de la Programmation mais simplement d'en mettre en valeur quelques unes. Pour cela, j'avais choisi comme fil conducteur la conception et l'utilisation de langages des programmation, domaine pour lequel j'ai tenté d'évaluer l'influence passée ou potentielle de la théorie. je me suis de plus limité aus langages de programmation séquentiels, en laissant de coté l'ensemble des travaux concernant le parallélisme.

Pourquoi une Théorie de la Programmation? Cet exposé étant destiné à un public assez large, on me pardonnera de le commencer en discutant le terme même de "programmation" et en justifiant le fait qu'elle possède une théorie.
L'activité de programmation est souvent vue, hors du cercle des spécialistes de l'Informatique et même parfois à l'intérieur, comme une activité subalterne. On la réduit à un simple codage d'algorithmes dans un jargon (le langage de programmation) qui n'a aucun intérêt en soi mais par lequel il faut bien passer pour faire exécuter aux machines ce qu'on attend d'elles. De ce point de vue, tous les langages paraissent à peu près équivalents et la création de nouveaux langages (lubie des informaticiens) est vue comme une simple perte de temps.
Bien sûr, ce point de vue ne résiste pas à un examen attentif. En particulier, si l'on veut bien observer le coût du développement de logiciel et

si on le compare au coût du matériel qui est en baisse constante, on doit bien constater que la programmation recèle des difficultés que quelques décennies d'efforts n'ont pas véritablement aplanies. Il y a quelque chose d'apparemment paradoxal dans le fait que le logiciel, entité immatérielle et duplicable à l'infini, ne puisse pas bénéficier d'une réduction des coûts au moins équivalente à celle que l'on obtient pour le matériel par la production en grande série des composants.
Le plus souvent, lorsqu'un programmeur écrit un module de programme, il peut être sûr que des milliers d'autres programmeurs ont écrit avant lui des modules de même spécification ou de spécification très proche. Pourquoi alors ne pas utiliser cet effort passé? Pourquoi la programmation ne peut-elle pas avoir un caractère plus cumulatif? Contrairement à ce que l'on pourrait penser, la raison n'est pas reliée en général aux problèmes de propriété du logiciel ni à l'existence d'une "Tour de Babel" des langages de programmation car ce problème se pose avec pratiquement la même acuité lorsque les logiciels sont dans le domaine public et écrits dans un même langage de programmation. Si les programmeurs ne peuvent "crassusser" autant qu'il serait souhaitable le travail de leur prédécesseurs (les programmeurs avisés le font cependant autant qu'il est possible), c'est qu'ils n'ont aucun moyen de connaitre les spécifications des programmes écrits par ces prédécesseurs et de les confronter à leurs propres spécifications. Si les spécifications pouvaient être exprimées de façon formelle et si l'on pouvait réaliser de façon formelle des tests d'équivalence de spécifications, alors on pourrait imaginer la constitution de véritables bibliothèques de programmes permettant de donner à l'activité de programmation un caractère cumulatif.
Ce problème des spécifications se pose avec la même acuité lorsqu'il s'agit de combiner entre eux des programmes existants pour réaliser des logiciels complexes. Il s'agit alors de savoir combiner les spécifications des composants pour produire une spécification pour le logiciel composé ou, d'un point de vue dual, de savoir décomposer une spécification en sous-spécifications.

Les objectifs de la théorie de la programmation
La Théorie de la Programmation a pour objectif, de mon point de vue, d'aller dans cette direction, c'est-à-dire de développer un cadre formel intégrant programmes et spécifications et permettant un développement des logiciels fondé sur la combinaison harmonieuse de programmes et de spécifications.

On peut objecter que derrière la notion de spécification formelle se cache toute la complexité des formules logiques et que les tests d'équivalence ou de compatibilité de spécifications impliquent des preuves mathématiques arbitrairement difficiles. C'est vrai. Pris dans toute sa généralité, ce programme rejoint celui des Mathématiques Constructives et l'on a d'ailleurs vu se développer dans la dernière décennie de remarquables convergences entre celles-ci et la Théorie de la Programmation. On peut cependant penser que dans une grande majorité de cas, les spécifications de logiciels ne font pas appel à des propriétés mathématiques difficiles et que même si la correction de certains composants des logiciels reposent sur des théorèmes difficiles, les difficultés liées à leur interconnexion ne viennent pas nécessairement de la difficulté du problème traité. Dans l'écriture d'un système de calcul formel ou d'un programme de gestion de bibliothèque, on trouvera des problèmes d'interconnection de modules qui sont semblables bien que la complexité intrins'que des problèmes traités soit de nature très différente. Par ailleurs, un programme ambitieux est souvent riche de retombées plus immédiates. Même si la Théorie de la Programmation est encore loin d'avoir rempli le programme très ambitieux esquissé ci-dessus, il me semble qu'elle a déjà apporté quelques contributions importantes à la conception de langages de programmation et à la pratique de la programmation, au moins pour ce qui concerne le développement de logiciels de recherche. Elle fournit d'autre part des critères pour juger de certaines innovations importantes introduites de façon pragmatique dans les langages de programmation.

Contenu de l'article

La suite de cette article est divisée en deux parties. La première partie est une tentative de panorama des avancées de la Théorie de la Programmation et en particulier du rôle qu'elle a joué dans la meilleur compréhension des concepts fondamentaux de la programmation, l'analyse des langages de programmation existants et la conception de nouveaux langages. La seconde partie est consacrée de façon plus détaillée à l'approche "fonctionnelle" de la programmation et à sa théorie. Ce choix est largement arbitraire et d'autres domaines de recherche esquissés mériteraient eux aussi un développement plus approfondi. Toutefois, nous pouvons en partie justifier notre choix par le fait que l'approche fonctionnelle de la programmation constitue sans doute un des domaines où les liens entre théorie et pratique sont les plus étroits.

1 La Théorie de la Programmation et son évolution

1.1 Le cheminement des idées en Théorie de la Programmation

L'interaction entre théorie et pratique

La Théorie de la Programmation doit éviter deux écueils. Elle ne doit pas pratiquer un suivisme systématique par rapport à l'évolution de la programmation réelle et se borner à essayer de formaliser les nouveaux concepts de programmation au fur et à mesure qu'ils apparaissent. Elle doit au contraire être capable d'émettre des jugements sur ceux-ci et éventuellement les rejeter même lorsqu'ils correspondent à une pratique très répandue. Mais elle ne peut pas non plus systématiquement ignorer les innovations introduites sur des bases pragmatiques même lorsque celles-ci ne semblent pas pouvoir s'intégrer dans le cadre conceptuel qu'elle a développé. Si la Théorie de la Programmation ne réussissait pas à intégrer certaines idées nouvelles ou à influencer les pratiques de programmation, elle serait condamnée à l'échec. Le cap n'est pas si facile à tenir. Certaines approches comme la programmation fonctionnelle ou la programmation logique qui sont en bonne partie issues de la recherche théorique n'ont pu commencer à faire la preuve de leur valeur que grâce aux progrès du matériel et il a fallu pendant longtemps une certaine foi visionnaire pour les défendre. D'autres, comme la programmation à objets, introduite sur des bases pragmatiques, ont largement pris les théoriciens à rebrousse-poil et il a fallu plusieurs années avant qu'ils puissent en dire quelque chose.
Notre point de vue est que ce qui distingue la théorie de la programmation de la logique constructive, même si les objectifs ultimes sont très proches, c'est précisément ce contact permanents avec les réalités informatiques et la capacité d'enrichissement mutuel de la pratique et de la théorie.

La vision des programmes comme objets formels

La Théorie de la Programmation est construite sur l'idée que les programmes sont des objets formels (des expressions), décomposés logiquement

en sous-expressions et sur lesquels on peut opérer des raisonnements, faire des calculs (par exemple de types) ou des opérations de transformations (optimisation, traduction, compilation) en d'autres termes des objets formels sur lesquels on peut effectuer des calculs symboliques. Pour que cette vision puisse être partagée par un grand nombre de programmeurs, il a fallu de nombreux progrès technologiques. Lorsque les programmes n'existaient concrêtement que sous la forme d'une pile de cartes perforées, il n'était guère facile de les concevoir comme des expressions formelles susceptible de manipulations logiques. Seuls de rares informaticiens comme par exemple les concepteurs d'Algol60 avaient cette vision à l'époque.
Avec l'accroissement des tailles des mémoires et les systèmes en temps partagés (à la fin des années soixante), on a vu apparaitre la notion d'éditeur. Pour les programmeurs, un programme devenait un objet manipulable à l'aide de l'ordinateur lui-même et l'idée qu'au delà de simples manipulations textuelles locales, on pouvait envisager des calculs plus complexes sur les programmes pouvait faire son chemin.
L'étape suivant a été l'introduction de la notion de "syntaxe abstraite" qui mérite une mention particulière. L'idée de distinguer entre apparence textuelle et structure profonde d'une expression est apparemment triviale et en tous cas aussi vieille que la notation algébrique en mathématique. Une expression telle que $x + (y \times z)$ peut également s'écrire $x + y \times z$ ou encore $x + yz$ et chacun passe mentalement au quotient en lisant de telle expressions même si nul ne sait comment ce quotient est exactement représenté dans notre cerveau. Dans les ordinateurs, la structure profonde (ou syntaxe abstraite) des expressions ou des programmes possède une forme arborescente représentée par une structure chaînée. Le langage Lisp a été le premier à permettre la manipulation de telles structures. On a appelé éditeurs structurés les éditeurs capables de travailler sur la structure profonde des expressions et qui ont peu à peu évolué vers des systèmes capables de calculs complexes sur les programmes. Par ailleurs, les langages de programmations ont peu à peu adopté des structures de données les dotant eux aussi de capacités de calculs symboliques et permettant aux idées de la théorie de la programmation de mettre ses concepts en pratique.

1.2 Quelques champs d'intervention de la Théorie de la Programmation

1.2.1 Programmation fonctionnelle et programmation Logique

Programmation fonctionnelle et programmation logique constituent des champs d'intervention particulièrement intéressants pour la théorie et on peut y observer toute la complexité des relations entre théorie et pratique en Informatique. Même si les langages fonctionnels et logiques ne connaissent pas une utilisation industrielle à très grande échelle, ils se sont néanmoins bien intégré au paysage informatique, au moins pour ce qui concerne les activités de prototypage et le développement de logiciels avancés et leur influence (celle de Lisp en particulier) sur les progrès de l'Informatique a été très importante.

Les deux familles de langages trouvent leur origine dans des travaux de logiciens (λ-calcul et résolution en logique du premier ordre) mais ont été conçues en vue d'applications (intelligence artificielle et analyse syntaxique complexe) et leurs créateurs et défenseurs les plus ardents ont toujours eu à coeur de se démarquer nettement des références théoriques. On pourrait voir dans cette attitude une négation de l'influence de la théorie sur le développement de ces langages mais il nous semble que ce serait une erreur. En paraphrasant ce que Marcel Proust disait du roman, on pourrait dire qu'un langage de programmation dans lequel on voit (trop) la théorie, c'est comme un cadeau sur lequel on aurait laissé le prix. Les apports théoriques réussis à un langage de programmation ne sont pas ceux qui oblige le programmeur à apprendre la théorie pour l'utiliser mais ceux qui s'imposent à lui par l' élégance et la simplicité des constructions auxquelles ils ont abouti.

Dans le cas de Lisp, l'influence théorique a été plutôt faible sur l'évolution ultérieure du langage bien que l'adoption de la portée lexicale des identificateurs, le traitement correct de la fonctionnalité et l'introduction de continuations dans des dialectes récents tels que Scheme puisse être portés à son crédit. En fait, la théorie de la programmation doit être surtout créditée pour l'introduction de nouveaux langages fonctionnels dont l'apport est détaillé dans la deuxième partie de ce texte.

En ce qui concerne Prolog, dont le nom même fait référence à ses fondements

logiques, il est clair qu'une grande partie de ses apôtres voit dans cette référence le guide principal pour son évolution. Le traitement de la négation, qui est essentiel pour les application à la représentation de connaissances et aux bases de données [26], repose sur des constructions sémantiques délicates et l'innovation récente la plus riche sans doute d'applications potentielles à savoir l'extension du principe d'unification de Robinson vers une notion de résolution de contraintes [27,16] sur des domaines particuliers est issue de travaux de théoriciens [42,68].
Enfin, on peut mentioner le problème de trouver un modèle de programmation réunissant les aspects fonctionnels et logiques, pour lequel il est maintenant clair que la solution ne peut venir que d'avancées théoriques importantes.

1.2.2 Typage et Modularité des Langages de Programmation

Origines de la notion de type
La notion de type a une double origine. En Logique, elle a été introduite par Russell en Théorie des Ensembles et par Church et Gödel dans le λ-calcul pour assurer la cohérence de ces formalismes comme formalismes de fondements des Mathématiques. Leur introduction a été motivée par la mise en évidence de paradoxes dans les versions non typées de ces théories.
En Informatique, la notion de type a été introduite dans le langage Fortran à la fin des années cinquante [70]. Il s'agisait d'assigner aux variables des expressions arithmétiques (dont l'usage constituait une innovation importante de Fortran) des types entier ou flottant de façon à pouvoir compiler les opérateurs en instructions entières ou flottantes de la machine.
A priori, les deux motivations sont si différentes qu'on a peine à comprendre que les deux notions méritent d'être associées. C'est précisément, à nos yeux, un des apports récents les plus importants de la Théorie de la Programmation que d'avoir mis en évidence les rapports profonds entre les deux notions et permis d'enrichir de façon considérable les structures de types des langages de programmations.
Nous renvoyons le lecteur à la deuxième partie de ce texte pour une étude plus détaillée de la notion de type car c'est dans le cadre fonctionnel que l'essentiel des travaux ont été faits. Nous mentionnerons simplement ici en quoi certaines innovations introduites dans le cadre fonctionnel peuvent

bénéficier à court terme à la programmation en général.
Même la simple notion de type introduite par Fortran est bien plus qu'une indication apportée au compilateur. Ce qui est en jeu, c'est la cohérence du programme. Un programme dont le typage est correct a une propriété forte: au cours de toutes ses exécutions possibles, aucune application d'un opérateur à des arguments erronés ne peut se produire. Cette cohérence est en quelques sorte le degré 0 de la correction des programmes et une condition préalable à tout espoir de preuve de correction vis-à-vis de spécifications. Nous pouvons donc considérer le problème du typage comme l'étape première et incontournable du programme que nous avons esquissé plus haut pour le théorie de la programmation.
D'un point de vue plus terre à terre, une vérification de types effectuée à la compilation permet de détecter la plupart des erreurs de programmation et facilite de façon considérable la mise au point. C'est la raison pour laquelle des langages comme Pascal, Ada ou C après quelques hésitations, ont adopté le principe du typage. Aucun n'est allé véritablement jusqu'au bout et le typage des programmes écrits dans ces langages ne garantit pas véritablement leur cohérence. Dans le cas de Pascal, il s'agit d'une lacune liée au typage des arguments fonctionnels. Dans le cas de C, il s'agit d'un laxisme plus délibéré: lorsqu'on utilise un langage de programmation pour écrire des systèmes opérant sur des machines aux données non typées, il n'est pas possible de s'assujétir complêtement à une discipline de type. Cependant, même si cet argument est recevable, il est important que le système de vérification de type sache au moins distinguer entre les programmes véritablement cohérents et les autres (le système lint améliore C dans cette direction).
Dans ce cadre de la vérification de types, les apports récents sont le polymorphisme et la synthèse automatique qui viennent des langages fonctionnels mais qui pourraient être adaptés à d'autres style de programmation. Le polymorphisme est la possibilité pour un même programme d'avoir plus d'un type. Un algorithme permettant de trier des tableaux (mettons quicksort) s'exprime indépendemment de la nature (du type) des éléments du tableau et peut s'utiliser avec une infinité de fonctions de comparaisons différentes entre éléments. Si la conséquence du typage est d'obliger le programmeur à écrire autant de versions de quicksort qu'il veut faire d'applications de cet algorithme (comme c'est le cas en Pascal), la contrainte est

insupportable. Le polymorphisme résoud ce problème par l'introduction de variables de types et y ajoute un mécanisme de synthèse automatique qui permet au compilateur de trouver le type le plus général d'un programme sans intervention du programmeur.

Types abstraits

Un autre problème lié au typage est l'abstraction dont la motivation principale est liée aux problèmes de modularité. Lorsqu'on réalise un logiciel par assemblage de modules, chaque module doit inclure dans sa spécification une certaine vision du monde extérieur, constitué pour lui des modules avec lesquels il aura à communiquer. Pour simplifier au maximum cette spécification et aussi lui assurer la plus grande généralité possible, il convient de ne faire au sujet de ce monde extérieur que les hypothèses minimales nécessaires à la correction du modules et son insertion dans l'ensemble du système.

Si on veut réaliser un programme utilisant un dictionnaire à clés ordonnées, on pourra s'appuyer sur un module indépendant dans lequel sont définies une structure de dictionnaire et des fonctions d'insertion et de recherche. La façon dont ce module est réalisé et en particulier la structure concrête du type dictionnaire n'a pas à être prise en compte par le programme utilisateur. Seul compte le fait que le module dictionnaire fournit des fonctions d'insertion et de recherche ayant les propriétés attendues. Ce souci de généralité et de minimalité des hypothèses est clairement de nature algébrique et a donné lieu au développement de la théorie des Types Abstraits Algébriques [72]. Bien que les chercheurs suivant cette approche se soient avant tout intéressé aux problèmes de spécification et de méthodologie de la programmation [50,51], ils ont également développé des langages tels que CLU et exercé une influence une influence non négligeable sur le développement du système de modules de nombreux autres langages récents.

1.2.3 Synthèse automatique d'outils

Un autre champ d'intervention très important de la théorie de la programmation est la synthèse automatique d'outils à partir de spécifications formelles. L'idée est de partir d'une spécification formelle d'un langage de programmation, incluant la description formelle de sa syntaxe et de sa sé-

mantique et de synthétiser automatiquement des logiciels permettant l'utilisation de ce langage tels que analyseur syntaxique, éditeur, interpréteur ou compilateur.
L'aspect syntaxique de ce programme a été réalisé depuis longtemps grâce aux apports de la Théorie des Langages et popularisé par des utilitaires tels que LEX et YACC sous UNIX.
L'aspect sémantique est encore du domaine expérimental mais on peut mentionner dans ce domaine quelques remarquables réussites qui s'appuient sur différents formalismes sémantiques comme la sémantique dénotationnelle, la sémantique opérationelle définie sous forme de règles d'inférence [9,15,41] ou les attributs sémantiques [66,40]. On peut raisonnablement penser que de tels systèmes seront utilisés dans l'avenir dans la phase de conception des langages de programmation et permettront de tester en vraie grandeur ces langages avant que leur définition ne soit figée et que des compilateurs industriels ne soient écrits. On peut noter qu'une d'efinition formelle du langage Ada avait été donnée à l'epoque de la conception de ce langage mais que les systèmes d'exécution disponibles à l'époque n'avait pas permis une exploitation effective de cette définition.

1.2.4 Vérification et analyse de programmes

La notion de preuves de programmes a donné lieu à un très grand nombre de travaux notamment dans les années 70 [21]. S'appuyant sur la logique du premier ordre et prenant en compte pour l'essentiel des langages de programmation traditionnels de type Pascal, ils n'ont pas débouché sur des systèmes effectifs largement utilisés. Même dans le cadre fonctionnel, a priori plus adapté aux preuves formelles, des succès ponctuels comme le système de Boyer et Moore [10] et le système LCF [63] n'ont pas connu une très large utilisation. Il semble bien que l'espoir de réaliser des systèmes de preuves ambitieux pour des langages de programmation existants qui n'ont pas été conçu dans ce but a peu de chance d'aboutir. Il est donc nécessaire de s'orienter vers des approches où programmes, spécifications et preuves sont intégrés au départ dans un formalisme unifié. Ce type d'approche est développé dans la seconde partie de cette article.
Toutefois, cet effort a eu tout de même des retombées intéressantes. Si on limite son ambition à la détection de propriétés simples pouvant aboutir par

exemple à l'optimisation de la compilation ou à la détection de possibilités de parallélisme. C'est le cas notamment des techniques d'analyse de programmes reposant sur les notions d'interprétation abstraite et d'évaluation partielle [22,61] qui suscitent actuellement un grand intérêt [1].

2 Les apports récents du point de vue fonctionnel

Le point de vue fonctionnel sur la programmation trouve son origine et ses méthodes dans le λ-calcul, formalisme logique introduit par Church vers 1930 dans une perspective de fondement constructif des Mathématiques et basé sur un formalisme fonctionnel effectif indépendant de la Théorie des Ensembles.
Si les résultats de Gödel ont ruiné le le programme de Hilbert visant à la démonstration par les Mathématiques de leur propre cohérence et, du même coup, la motivation d'origine des travaux de Church, le λ-calcul a gardé un double intérêt. D'une part, il est un des formalisme permettant de définir la notion de fonction calculable (l'équivalence avec les machine de Turing a été montrée par Kleene) et d'autre part, il permet de coder les preuves des différents formalismes introduits par la Théorie de la Démonstration, discipline qui s'et attachée après Gödel et grâce notamment aux travaux de Gentzen, à étudier la puissance relative de différents systèmes logiques. Ce codage des preuves dans le λ-calcul fait apparaitre une complête analogie entre formules logiques et types du λ-calcul désignée sous le nom d'isomorphisme de Curry-Howard" [19,32].
L'intérêt des informaticiens pour le λ-calcul s'est tout d'abord porté sur ses aspects calculatoires. On peut en effet y voir un langage de programmation à la fois très simple et très riche de propriétés mathématiques. Il constitue l'ancêtre et le noyau de tous les langages fonctionnels actuels. Non seulement il a permis de discuter dans un formalisme bien adapté de problèmes très concrets de l'informatique comme la portée des identificateurs ou de modes de transmission de paramètres mais aussi a servi de base aux principaux travaux concernant la définition de l'ensemble de la sémantique de langages de programmation. Dans ce domaine, le pionnier a été Landin qui a fournit dans les années soixante une traduction en λ-calcul du langage

Algol [47], proposé une notation (ISWIM [48]) qui est à la base des langages fonctionnels modernes et conçu une machine virtuelle (SECD) pour leur implantation [46].
Plus récemment et à la suite des travaux de DeBruijn [24], Girard [28,29]et Martin-Löf [53], des informaticiens comme Constable [18], Coquand et Huet [38], Hayashi [31] , se sont intéressés à l'implantation effective de systèmes de preuves basés sur le λ-calcul et à l'utilisation de l'isomorphisme de Curry-Howard comme principe unificateur entre les notions de types et de spécification en programmation. L'idée fondamentale qui sous-tend cette approche est que partant d'une spécification vue comme une formule logique, la preuve constructive de la satisfiabilité de cette spécification représentée par un terme du λ-calcul est un programme répondant à la spécification ou du moins contient en un certain sens un tel programme. Les objectifs de la Théorie de la Programmation reçoivent ainsi une formulation très concrête.

La suite de cette section est divisée en deux parties. La première est consacrée aux apports concrets du point de vue fonctionnel tels qu'ils se présentent dans des langages existants et diffusés. La seconde est consacrés aux aspects plus futuristes inspirés par l'isomorphisme de Curry-Howard.

2.1 Principes et Evolution des Langages Fonctionnels

Les langages de programmation traditionnels combinent les notions d'expressions (entités susceptible d'être évaluées en une valeur) et commandes (ordres destinés à modifier l'état courant). Cette dichotomie se traduit par l'existence de deux constructions distinctes, les fonctions et les procédures, pour construire des entités nommées et paramétrées correspondant à des expressions ou à des commandes complexes.
Les langages fonctionnels retiennent eux les seules notions d'expression et de fonction. Une exécution y est toujours vue comme une évaluation d'expression et non comme l'exécution d'une suite d'instructions. Les fonctions y correspondent exactement à la notion mathématique de fonction. En cela, ils ne font d'une certaine façon que mener jusqu'au bout l'évolution des langages de programmation commencé avec Fortran et qui a consisté essentiellement en un effort pour échapper aux notions héritées de la structure des machines et évoluer vers des notations plus mathématiques, mieux adaptées

à la formulation des problèmes. Notons que le créateur de Fortran s'est fait le défenseur de ce point de vue [5].
On peut remarquer que le rapport des langages fonctionnels aux langages machines est exactement celui du λ-calcul aux machines de Turing. De ces deux formalismes équivalents de définition des fonctions caculables, le second est de loin celui qui est le plus populaire en raison sans doute de ce qu'il apparait plus concret, plus proche de la réalité des machines. Du point de vue de la programmation, le premier est infiniment plus intéressant par ses constructions de haut niveau et son contenu logique. Il a toutefois fallu plusieurs décennies de progrès dans la technologie des processeurs et celle des compilateurs pour qu'il apparaisse enfin réaliste.

2.1.1 Apports des Langages Fonctionnels

Fonctions d'ordre supérieur
La première caractéristique des langages fonctionnels est bien sûr le statut des fonctions. Celles-ci sont notée de façon très proche de la notation mathématique usuelle. Par exemple
`fun (x,y) -> x+2*y`
est la façon dont le langage CAML note la fonction qui à x et y associe $x + 2y$.
Etant considérées comme des valeurs à part entière, les fonctions peuvent être passées en arguments à d'autres fonctions ou rendues en résultat. On parle alors de fonctions d'ordre supérieur. Par exemple, la composition de fonctions, au sens mathématique s'écrira
`fun (f,g) -> (fun x -> f(g(x)))`
La possibilité d'écrire de telles fonctions dans le langage accroit de façon considérable sa puissance d'expression en donnant des possibilités de paramétrisation qui n'existent pas ou n'existent qu'à travers des constructions beaucoup plus compliquées dans les langages de programmation courants

Polymorphisme et synthèse automatique de types
De façon à assurer la cohérence de l'utilisation de telles fonctions et à assister le programmeur dans leur utilisation, le typage est absolument nécessaire. Par ailleurs, ce typage doit nécessairement être polymorphe. Par exemple, la notion mathématique de composition de fonction s'applique

à des fonctions `f` et `g` arbitraires pourvu que le codomaine de la seconde corresponde au domaine de la première.
Dans un langage fonctionnel, l'expression
`fun (f,g) -> (fun x -> f(g(x)))`
qui dénote la composition de fonction aura le type
$\forall\alpha\ \beta\ \gamma.\quad (\beta\ \rightarrow \gamma) \times (\alpha \rightarrow \beta)\ \rightarrow (\alpha\ \rightarrow\ \gamma)$
Ce type est le plus général possible pour la composition de fonction en ce sens qu'il ne retient que la contrainte minimale pesant sur les type des paramètres `f` et `g`.
Par ailleurs, il est synthétisé automatiquement par le compilateur. Notons que la synthèse automatique de types repose sur l'algorithme d'unification et se fait essentiellement par un calcul à la Prolog.

Types définis par l'uilisateur et appel par filtrage

Les constructions que nous allons décrire maintenant ont été introduites pour la première fois dans le langage Hope [11]. Les définitions récursives de types avec constructeurs ont sans doute été inspirées par les équations aux domaines de la sémantique dénotationnelle. L'appel par filtrage vient plutôt de l'Intelligence Artificielle et des travaux sur les systèmes de réécriture.
Soit à définir une structure de données correspondant à des arbres binaires. On pourra par exemple écrire:
`type 'a arbrebin = 'Vide | 'Bin of 'a * 'a arbrebin * 'a arbrebin`
Cette déclaration définit de façon inductive le type paramétré **arbrebin** (`'a` est la notation utilisée dans le langage CAML [71] pour les variables de type que nous avons notées jusqu'ici α, β, γ), comme contenant la valeur `'Vide` et toutes les valeurs `'Bin(a,t1,t2)` où a est une valeur de type `'a` et `t1` et `t2` sont des arbres binaires.
Les entités `'Vide` et `'Bin` sont appellées constructeurs du type arbrebin. Les fonctions utilisant le type arbrebin seront définies par cas sur la structure des arbres en distingant le cas `'Vide` et le cas `'Bin`. Par exemple, une fonction calculant l'image miroir d'un arbre binaire pourrait être définie par:

```
miroir('Vide) = 'Vide
miroir('Bin((a,t1,t2)) = 'Bin(a,miroir(t2), miroir(t1))
```

Une fonction prenant en argument un arbre binaire contenant des nombres et calculant la somme de ces nombres pourrait s'écrire:

```
somme('Vide)=0
somme(n,t1,t2)= n+somme(t1)+somme(t2)
```
La combinaison de telles définitions de types et de l'appel par filtrage permettent de définir très simplement des syntaxes abstraites et de programmer sur ces syntaxes. Elles sont à l'origine de la puissance des langages fonctionnels pour réaliser des systèmes de calcul symbolique.

Calcul sur des objets infinis

Le modèle de calcul des langages fonctionnels qui est fondé sur la réécriture d'expresssions est moins contraint que celui des langages traditionnels. En particulier, il n'exige pas que tous les arguments d'une fonction aient été calculés avant que celle-ci ne soit appliquée. Ceci permet en particulier de programmer conceptuellement avec des structures de données infinies sachant que les calculs n'en évalueront qu'une partie finie.
Par exemple, soit à calculer le n-ième nombre premier en utilisant le crible d'Eratosthène consistant à élaguer la suite des nombres entiers de ses éléments non premiers. La programmation de cet algorithme extrêmement simple est inutilement compliquée par le fait qu'il faut essayer de deviner la longueur de la sous-suite initiale des nombre entiers qui sera suffisante pour y trouver le n-ième nombre premier. Les langages fonctionnels permettent d'utiliser simplement la suite infinie des nombre entiers.

2.1.2 Implantation

L'élégance de la notation fonctionnelle serait de peu d'intérêt s'il n'était pas possible de la compiler de façon efficace. Heureusement, la dernière décennie a vu le développement de nombeuses implantations de langages fonctionnels (SML [55], CAML [71], LML [4], CLEAN , Miranda [67], Haskell [34]) dont certaines sont extrêmement efficaces.
Cette efficacité repose en partie sur des innovations techniques qui ne doivent rien à la théorie en particulier dans le domaine de la gestion de la mémoire mais il est tout-de-même remarquable que cet effort pratique de réalisation de compilateur a été souvent inspiré par des considérations de nature théorique et développées au sein d'équipes de théoriciens.
Tout d'abord, l'étude fine des règles de calculs sures ont fait l'objet de très nombreux travaux qui servent de références pour la correction des implan-

tations [6,8,49,62,69]. Ensuite, le processus même de la compilation de ces langages ont pu être décrits de façon fine à un niveau théorique [7,64,20]. Enfin notons que même si les résultats d'optimalité des règles de calculs [43,45] ne se sont pas traduits pour le moment en compilateurs optimaux en un sens pratique, ils n'en reste pas moins une référence importante et un espoir pour les développements futurs.

2.2 Représentation uniforme des Preuves et des Programmes

2.2.1 Identité entre types et formules logiques

Dans une vision formalisée des Mathématiques, les notions de propriété et de démonstration correspondent à des objets formels appelés formules et preuves. Il se trouve que pour les systèmes logiques développés àla suite de Gentzen par la Théorie de la Démonstration, le lambda-calcul fournit une notation parfaite pour les preuves.
On peut s'en rendre compte immédiatement en considérant un système logique minimal pour l'implication. On considérera ici des systèmes de Gentzen manipulant des séquents de la forme $H \vdash A$ qui représentent la validité d'une formule A sous les hypothèses H. Si nous nous limitons aux formules propositionnelles implicatives, c'est-à-dire celles qui s'écrivent uniquement à l'aide de variables propositionnelles et d'implications, les règles logiques sont au nombre de trois.

$$(TAUT) \qquad \frac{}{H, A \vdash A}$$

$$(INTRO) \qquad \frac{H, A \vdash B}{H \vdash A \Rightarrow B}$$

$$(ELIM) \qquad \frac{H \vdash A \Rightarrow B \quad H \vdash A}{H \vdash B}$$

Si l'on veut étudier les propriété d'un tel système (par exemple pour établir qu'il est décidable), il faut pouvoir caractériser l'ensemble des preuves d'une formule $H \vdash A$ et donc introduire une notation pour celle-ci. Pour cela, on nommera les hypothèses à l'aide de variables et on utilisera une syntaxe

pour noter les preuves des conclusions des règles INTRO) et (ELIM). Il se trouve que les constructions d'abstraction et d'application du λ-calcul fournissent une syntaxe idéale. Le système devient donc:

$$(TAUT) \qquad \frac{}{H, x : A \vdash x : A}$$

$$(INTRO) \qquad \frac{H, x : A \vdash M : B}{H \vdash \lambda x.M : A \Rightarrow B}$$

$$(ELIM) \qquad \frac{H \vdash M : A \Rightarrow B \quad H \vdash N : A}{H \vdash (MN) : B}$$

Rappelons que dans le λ-calcul, la notation $\lambda x.M$ représente intuitivement la fonction qui à x associe M et la notation (MN) correspond à l'application de M à N. Notre notation prend donc un sens constructif. Dans la règle (INTRO), la preuve $\lambda x.M$ de la formule $A \Rightarrow B$ peut être vue comme une fonction qui prend une preuve de A et fabrique une preuve de B. Réciproquement, dans la règle (ELIM), on obtient une preuve (MN) de B en appliquant la preuve M de $A \Rightarrow B$ à la preuve N de A.
On remarque également que cette vision est en accord avec la notion de types si l'on interprète $A \Rightarrow B$ comme le type des fonctions de A dans B. Une formule peut donc être vue de façon consistante comme le type de sa preuve. Réciproquement, les types simples des langages fonctionnels peuvent donc être vus comme des spécifications élémentaires: exactement celles que l'on peut écrire dans la logique propositionnelle implicative.
Il est tout à fait remarquable que cette identité entre types et spécifications s'étende à l'ensemble des constructions logiques. Nous renvoyons le lecteur à [19] pour une introduction très claire à cette construction.

2.2.2 Utilisation de cette approche en programmation

L'utilisation potentielle de cette identité en programmation est assez simple à comprendre. Si nous considérons une formule logique de la forme $\forall x.\exists y.A$ qui peut s'interpréter comme la spécification d'une programme qui à partir d'une donnée x devra calculer un y tel que $A(x, y)$, la preuve de la formule, représentée sous la forme d'un λ-terme fournira un moyen effectif de calculer y à partir de x.

Il faut cependant réaliser que ce λ-terme contiendra en général une information logique beaucoup plus complexe que son contenu proprement calculatoire. Lorsque l'on veut calculer le pgcd de deux nombres par l'algorithme d'Euclide, on ne souhaite pas normaliser au passage la preuve de l'algorithme. Il faut donc développer des méthodes pour extraire d'une preuve l'information calculatoire.
Par ailleurs, les λ-termes typés codant les preuves sont fortement normalisables c'est-à-dire que les programmes qu'on peut en extraire sont des programmes qui terminent toujours. La classe des programmes que l'on pourra obtenir est donc limitée par rapport à celles des programmes que l'on peut écrire dans les langages de programmation habituels. Bien qu'il soit possible de démontrer que la classe des fonctions totales qu'on obtient ainsi est extrêmement grande [44], cela n'implique pas que les programmes écrits ainsi puissent avoir la même complexité que leurs homologues écrits dans des langages classiques. Peu d'études ont encore été menées sur ce sujet [17] .
Notons par ailleurs que l'enrichissement de la notion de type par des constructions logiques telles que les quantifications universelles et existentielles a eu aussi des retombées plus immédiates. La quantification universelle a été utilisée pour étendre le polymorphisme des langages et prendre en compte la notion de sous-types, ouvrant entre autres la voie à une formalisation des langages à objets [58]. La quantification existentielle a été utilisée pour modéliser les notions de types abstraits et de modules [12,52,56],.

2.2.3 Perspectives de cette approche

L'identification entre spécifications et types que nous n'avons pu qu'esquisser ici, suggère une nouvelle approche de la programmation fondée sur la manipulation des spécifications, de preuves et de programmes dans un cadre unifié. Son application pose toutefois de nombreux problèmes. Tout d'abord, elle nécessite le développement de systèmes puissants d'aide à la preuve. Ensuite, elle nécessite le développementt de méthodes d'extraction de programmes à partir de preuves. Enfin elle interdit l'usage libre de la récursivité.
Son intérêt est cependant évident. D'une part elle donne une direction concrête de recherche permettant d'avancer vers ce que nous avons pré-

senté comme le projet fondamental de la Théorie de la Programmation: la manipulation dans un formalisme commun de programmes et de spécification. D'autre part, et de façon plus immédiate, Elle a indiqué des voies pour enrichir les types des langages de programmations actuels..

Références

[1] Abramsky S. and Hankin C. Eds., Abstract Interpretation of Declarative Languages, Ellis Horwood (1987).

[2] Appel A. and Mac Queen D., A standard ML compiler, Conference on Functional Programming Languages and Computer Architecture (1987), Springer LNCS 274.

[3] Apt K., Logic Programming, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[4] Augustsson L. A compiler for Lazy ML, ACM Conference on Lisp and Funstional Programming (1984)

[5] Backus J., Can programming be liberated from the Von Neumann style?, CACM 21, 8 (1978).

[6] Barendregt H., The Lambda Calculus: its Syntax and Semantics, North Holland (1980).

[7] Barendregt H., Functional Programming and Lambda Calculus, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[8] Berry G. and lévy JJ., Minimal and optimal computation of recursive programs, JACM 26, 1 (1979).

[9] Borras P., Clément D., Despeyroux Th., Incerpi J., Kahn G., Lang B., Pascual V., Centaur: the system, Proc. ACM Software Eng. Symp. on Partical Software Development Environments (SIGSOFT 88).

[10] Boyer R. and Moore J., A Computational Logic, Academic Press (1979).

[11] Burstall R., MacQueen D. and Sannella D., Hope: an experimental applicative language, Lisp Conference (1980)

[12] Cardelli L. and Wegner, On undertanding types, data abstraction and polymorphism, ACM Computing Surveys, 17 4 (1985).

[13] Cardelli L., Basic Polymorphic Type Checking, Science of Computer Programming, 8, 2 (1987)

[14] Cardelli L. and Mitchell J., Operations on records, Conference on Mathematical Foundations of Programming Languages Semantics, New-Orleans (1989)

[15] Clément D., Despeyroux J., Despeyroux T., and Kahn G., Natural Semantics on The Computer, Rapport INRIA 416.

[16] Colmerauer A., Opening the Prolog III Universe, BYTE Magazine 12, 9 (1987).

[17] Colson L., About primitive recursive algirithms, Sicteenth International Symposium on Automata, Languges and Programming (1989), Springer-Verlag.

[18] Constable R. et al., Implementing Mathematics in the NuPrl System, Prentice-Hall (1986).

[19] Coquand T. On the analogy between propositions and types, in Logical Foundations of Functional Programming, G.Huet Ed., Addison-Wesley, 1990.

[20] Cousineau G., Curien P-L. and Mauny M. The Categorical Abstract Machine, Science of Computer Programming 8 (1987)

[21] Cousot P., Methods and Logics for Proving Programs, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[22] Cousot P., Demantic Foundations of Program Analysis, in Muchnick S. and Jones N. Eds., Program Flow Analysis: Theory and Applications, Prentice Hall (1981).

[23] Curien P-L. An abstract framework for environment machines, à paraître dans Theoretical Computer Science (1991).

[24] de Bruijn N., A survey of the project Automath, in To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press (1980).

[25] Dershowitz N. and Jouannaud J-P., Rewrite Systems, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[26] Gallaire H., Minker J. and Nicolas J-M., Logic and Databases: A deductive Approach, ACM Computing Surveys 1984 , 2.

[27] Gallaire H., Logic Programming: Further Devellopments, Symposium on Logic Programming, Boston (1985).

[28] Girard J-Y., Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur, Thèse d'Etat, Université Paris VII (1972)

[29] Girard J-Y., The system F of variable types, 15 years later, in Logical Foundations of Functional Programming, G.Huet Ed., Addison-Wesley, 1990.

[30] Gunter C. and Scott D., Semantic Domains, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[31] Hayashi S., An introduction to PX, in Logical Foundations of Functional Programming, G.Huet Ed., Addison-Wesley, 1990.

[32] Howard W., The formula-as-types notion of constructions, in To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press (1980).

[33] Hudak P., The Conception, Evolution and Application of Functional Languages, ACM Computing Surveys, 1989, 3.

[34] Hudak P., Hudak P. and Wadler P. Eds., Report on the Functional Programming Language Haskell, Technical Report YALEU/DCS/RR656, Yale University (1988).

[35] Huet G. Ed., Logical Foundations of Functional Programming, University of Texas Year of Programming Series, Addison-Wesley, 1990.

[36] Huet G., A uniform approach to type theory, in Logical Foundations of Functional Programming, G.Huet Ed., Addison-Wesley, 1990.

[37] Huet G., The calculus of Constructions: documentation and users's guide, Rapport INRIA 110 (1989)

[38] Huet G. and Lévy JJ., Call by Need Computations in Non-Ambiguous Linear Term Rewritig Systems, Rapport INRIQ 513 (1986).

[39] Johnsson T, Implementation of Lazy Functional Languages, Prog. Meth. Group Report 53, Chalmers University (1988).

[40] Jourdan M. et Parigot D., The FNC-2 System User's Guide and Manual, INRIA (1987).

[41] Kahn G., Natural Semantics, in Programming of Future Generation Computers, Fuchi and Nivat Eds, Elsevier (1988).

[42] Jaffar J. and Lassez J-L., Constraint Logic rogramming, Symposium on Principles of Programming Languages, Munich (1987).

[43] Kathail V. Optimal Interpretors for Lambda Calculus, PhD Thesis, MIT, 1990.

[44] Krivine J-L., Lambda Calcul: types et modèles, Masson (1990).

[45] Lamping J., An algorithm for optimal lambda calculus reduction, Symposium on Principles of Programming Languages (1990).

[46] Landin J., The mechanical evaluation of expressions, Computer Journal 6, 4 (1964)

[47] Landin J., A correspondence between Algol60 and Church's lambda notation, CACM 8 (1985).

[48] Landin J., The next 700 programming languages, CACM 9, 3 (1966).

[49] Lévy JJ., Optimal reductions in the lambda calculus, in To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press (1980).

[50] Liskov B. and Guttag J., Abstraction and Specification in Program Development, MIT Press (1986)

[51] Liskov B. and Zilles S., Programming with Abstract Data Types, ACM SIGPLAN Notices 9 (1974).

[52] MacQueen D., Using dependant types to express modular structures, Proc. Symposium on Principles of Programming Languages (1885)

[53] Martin-Lóf P., Intuitionistic Type Theory, Studies in Proof Theory, Bibliopolis, 1980.

[54] Milner R., A theory of type polymorphism in programming, J. Computer and System Sciences, 17, 3 (1978).

[55] Harper R., MacQueen D. and Milner R., Standard ML, ECS-LFCS-86-2, Laboratory for Computer Science, Univ. of Edinburgh (1986).

[56] Mitchell J. and Plotkin G., Abstract data types have existential types, Symposium on Logics in Computer Science (1987)

[57] Mitchell J., Type systems for Programming Languages, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[58] Mitchell J., Towards a typed foundation for method specialization and inheritane, Proc. Symposium on Principles of Programming Languages (1990)

[59] Paulin-Mohring C., Extracting Fω programs from proofs in the Calculus of Constructions, Proc. Symposium on Principles of Programming Languages (1989)

[60] Mosses P., Denotational Semantics, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

[61] Muchnick S. and Jones N. Eds., Program Flow Analysis: Theory and Applications, Prentice Hall (1981).

[62] Nivat M., On the Interpretation of Recursive program Schemes, Symposia Matematica Vol XV (1975).

[63] Paulson L., Logic and Computation: Interactive proof with Cambridge LCF, Cambridge University Press (1987)

[64] Peyton-Jones S., The Implementation of Functional Programming Languages, Prentice Hall (1987).

[65] Sethi R., Programming Languages Concepts and Constructs, Addison Wesley (1989).

[66] Teitelbaum T. and reps T., The Cornell Program Synthetizer: a syntax directed programming environment, CACM 24 (1981).

[67] Turner D., Miranda: a non strict functional language with polymorphic types, in Funstional Languages and Computer Architecture, Springer-Verlag LNCS 201 (1985).

[68] Van Hentenryck P., Constraint satisfaction in Logic Programming, MIT Press (1989).

[69] Vuillemin J., Syntaxe, Semantique et Axiomatique d'un Langage de Programmation Simple, Thèse d'Etat de l'Université Paris VII (1974).

[70] Wexelblat R., History of Programming Languages, Academic Press (1981).

[71] Weis P. et al., The CAML Reference Manual, Rapport INRIA 121 (1990)

[72] Wirsing M.. Algebraic Specifications, in Hanbook for Theoretical Computer Science, Van Leeuven Ed., North-Holland.

RUBRIQUE LIVRES

LIVRES PROPOSÉS A SPECIF

Cette rubrique propose des ouvrages récents dont Specif a eu connaissance. Il ne s'agit pas de commentaires, mais simplement de la "quatrième de couverture". N'hésitez pas à donner votre point de vue sur son utilité. Si elle vous paraît intéressante, aidez nous à la mettre à jour.

Thésèse ACCART HARDIN, Véronique DONZEAU-GOUGE VIGUIE, *Concepts et outils de programmation, le style fonctionnel, le style impératif avec CAML et Ada*, 581 pages, InterEditions. Bien que s'adressant à des débutants, les auteurs de ce livre ont délibérément choisi d'introduire leurs lecteurs aux concepts et aux outils de programmation les plus avancés.

L'ouvrage commence par une approche fonctionnelle de la programmation, relayée par l'apprentissage des langages impératifs, et montre comment passer harmonieusement d'un texte écrit dans le style fonctionnel à un texte écrit dans un langage de programmation «classique».

Les langages choisis comme supports de démonstration sont le langage fonctionnel CAML et le langage impératif Ada. De nombreux exemples et exercices clarifient les approches théoriques et permettent au lecteur de tester sa compréhension. Toutes les solutions sont fournies.

Ce cours repose sur l'expérience pédagogique des auteurs en premier cycle du CNAM.

Douglas COMER, *TCP/IP: Architecture, protocoles, applications*, 575 pages, InterEditions. Cet ouvrage est destiné principalement à tous ceux qui ont besoin d'aborder et de comprendre les protocoles TCP/IP (Transmission Control Protocol/Internet Protocol) largement utilisés dans le monde des communications. Mais l'explication des principes généraux de la communication entre calculateurs — architecture, couches, multiplexage, encapsulation, adressage, routage et nommage — et les exemples spécifiques extraits de la famille des protocoles TCP/IP en font un ouvrage bien adapté à l'apprentissage des protocoles en général.

Passant des réseaux physiques aux applications, l'auteur utilise une approche ascendante et décrit TCP/IP comme un mécanisme de communications interprocessus où les notions d'interconnexion et d'interopérabilité constituent des idées de base. Le livre forme un bon point de départ pour aborder l'interconnexions des systèmes ouverts (modèle OSI) en donnant une synthèse qu'il est souvent difficile d'acquérir par la seule lecture des normes. Les exercices de fin de chapitre permettent de vérifier l'assimilation des concepts; les références bibliographiques et les annexes font de l'ouvrage un guide complet sur le sujet.

A la fois support de cours et ouvrage de référence professionnel, celui-ci intéressera les étudiants en informatique, les ingénieurs systèmes et les architectes réseaux. Il leur apportera une analyse approfondie de la technique TCP/IP, les aidera à choisir les produits du marché et à maîtriser la conception, l'administration et l'utilisation d'interconnexions de réseaux.

Oliver JONES, *Le système X Window*, 602 pages, InterEditions. A l'heure où l'évolution vers les systèmes ouverts préoccupe une bonne partie de l'industrie informatique, le système de fenêtrage X Window, solidement établi sur des bases techniques éprouvées, connaît un essor exceptionnel, encore amplifié par le succès d'interfaces graphiques (Motif, Open Look,...) élaborées à partir de cette norme.

Cet ouvrage, déjà reconnu comme référence dans sa version originale, met à la portée des programmeurs les concepts particulièrement riches de ce système de fenêtrage portable massivement implanté dans le monde Unix. Pour analyser ces concepts, l'auteur adopte une démarche progressive s'appuyant sur un exemple servant de fil conducteur. Fenêtres, graphisme, affichage de textes et gestion de la couleur, opérations sur les trames de points, gestion de la souris et du clavier sont ainsi décrits en détail. L'ouvrage se termine par des notions plus complexes — traitement avancé des événements et communications entre

programmes. L'édition française comporte une bibliographie, un glossaire et trois annexes sur les interfaces graphiques qui viennent compléter les sept annexes de l'édition originale.

Ce livre se révèlera précieux aussi bien pour les développeurs ayant le souci de bâtir des applications tirant pleinement parti des possibilités de leurs stations de travail que pour les spécialistes élaborant des interfaces, boîtes à outils ou autres logiciels spécifiques.

Bertrand MEYER, *Introduction à la théorie des langages de programmation*, 451 pages, InterEditions. Pour les développeurs de logiciel, les langages de programmation sont l'outil fondamental et quotidien. Trop peu pourtant connaissent les bases théoriques qui permettent de mieux maîtriser les langages et de mieux les employer.

Par une démarche progressive, claire et bien structurée, l'auteur présente successivement les notions de syntaxe abstraite et de sémantique formelle. Puis il développe la sémantique dénotationnelle et la sémantique axiomatique. Pour illustrer son discours, il expose et approfondit des concepts importants, comme le lambda-calcul et les définitions récusrsives.

Ce livre fournit, de surcroît, une excellente étude comparative des caractéristiques de différents langages de programmation. Algol, Ada, Lisp, Pascal, C... et les langages à objets, notamment Eiffel, sont à leur tour évoqués.

Il en ressort une vue synthétique très riche de leurs différences et/ou de leurs parentés ainsi que des choix qui ont présidés à leur implémentation.

La présentation a été conçue pour des praticiens de la programmation; l'auteur utilise en permancence des exemples de programmes concrets et des analogies empruntées aux situations les plus courantes de la construction de logiciel. Tous les concepts mathématiques nécessaires sont empruntés à la théorie élémentaire des ensembles et définis clairement dans un bref chapitre d'introduction. Enfin, le lecteur appréciera de pouvoir évaluer ses acquis grâce aux exercices proposés à la fin de chaque chapitre.

Les concepteurs de langage puiseront dans cet ouvrage les bases fondamentales de leur travail. Les étudiants et les ingénieurs y trouveront un exposé didactique leur permettant d'approfondir leurs connaissances théoriques.

DIVERS

APPEL AUX COMMUNICATIONS

CFIP'93

Colloque Francophone sur l'Ingénierie des Protocoles

7-9 Septembre 1993, Montréal, Canada

Instructions aux Auteurs

Si vous désirez soumettre une communication, veuillez envoyer **cinq exemplaires** de l'article complet, avant le 15 février 1991, à:

Rachida Dssouli (Université de Montréal, Canada)

Université de Montréal
Faculté des arts et des sciences
Département d'informatique et
de recherche opérationnelle
C.P. 6128, Succursale A
Montréal, (Quebec)
H3C 3J7
Tel. : (514) 343 7599
email: dssouli@iro.umontreal.ca
Fax: (514) 343 5834

Les décisions du comité de programme vous seront notifiées le 15 avril 1991. Les versions finales des articles doivent parvenir avant le 15 juin 1991, pour être publiées dans les actes.

Dates à retenir

15 février: Date limite pour la réception des articles soumis.
15 avril 1991: Notification aux auteurs de la décision du comité de programme.
15 juin 1991: Date limite pour la réception des textes définitifs à inclure dans les actes du colloque.

APPEL AUX COMMUNICATIONS

CFIP'93

Colloque Francophone sur l'Ingénierie des Protocoles
7-9 Septembre 1993, Montréal, Canada

Comité de Programme

L'idée du Colloque Francophone sur l'Ingénierie des Protocoles qui a déjà eu lieu par deux fois (1988 et 1991), est de permettre à la communauté francophone des enseignants, chercheurs et industriels dans le domaine des protocoles et des réseaux informatiques, de faire régulièrement le point en langue française.

Le comité de programme souhaite se voir soumettre des communications qui traitent de tout sujet relatif au développement des protocoles de communication. La liste non exhaustive qui suit, donne quelques points qui peuvent être traités:

- Définition, choix et combinaison de protocoles pour des applications données;
- Conception, mise en oeuvre et gestion de réseaux et d'architectures de communication;
- Techniques et langages de spécification; outils d'analyse;
- Techniques et outils de vérification et de simulation; application des techniques d'intelligence artificielle;
- Techniques d'implantation et d'intégration des logiciels de communication dans les systèmes existants;
- Architectures, méthodes et outils de test de conformité et d'interopérabilité; mesures de performance;
- Méthodes formelles ou pragmatiques couvrant l'ensemble des étapes de développement des protocoles;
- Conception et développement de protocoles à haut débit et multimédia
- OSI, ISDN, ATM, ODP, Mobiles, ...

Des communications sont souhaitées de la part d'universitaires de chercheurs et d'industriels. Les articles de synthèse didactiques, les exposés de travaux de recherches théoriques ou d'expériences pratiques, les présentations de produits et les publications émanant de jeunes chercheurs sont les bienvenus.

Les textes doivent être écrits en français et la présentation orale doit être effectuée en français.

CALL FOR PAPERS

Université de Pau, France

6th International Workshop on Protocol Test Systems

Pau, France, September 28-30, 1993
Sponsored by IFIP TC6

IFIP

Program Chairman :

Omar Rafiq (Université de Pau, France)

Program Committee :

Gregor von Bochmann	*Université de Montréal, Canada*
Gérard Bonnes	*IBM, France*
Ed Brinksma	*University of Twente, The Netherlands*
Richard Castanet	*LaBRI, France*
Jan de Meer	*GMD-FOKUS, Germany*
Rachida Dssouli	*Université de Montréal, Canada*
Jean-Philippe Favreau	*NIST, USA*
Roland Groz	*CNET, France*
Michel Haulard La Brière	*SEPT, France*
Dieter Hogrefe	*University of Bern, Switzerland*
Jan Kroon	*PTT Research, The Netherlands*
Guy Leduc	*Université de Liège, Belgium*
Ching-Sung Lu	*Telecommunication Laboratories, Republic of China*
José Mañas	*Technical University of Madrid, Spain*
Pierpaolo Marchese	*CSELT, Italy*
Raymond E. Miller	*University of Maryland, USA*
Marc Phalippou	*CNET, France*
Behçet Sarikaya	*Bilkent University, Turkey*
Deepinder P. Sidhu	*University of Maryland, USA*
Roger Tarazi	*TEKELEC, USA*
Jan Tretmans	*University of Twente, The Netherlands*
Hasan Ural	*University of Ottawa, Canada*
Umit Uyar	*AT&T Bell Laboratories, USA*
Son T. Vuong	*University of British Columbia, Canada*
Yoshihiko Yokoyama	*INTAP, Japan*

Aims

IFIP will sponsor the sixth International Workshop on Protocol Test Systems (IWPTS) end of September 1993 in Pau (France). This workshop is dedicated to bringing together researchers and practitioners interested in the development and use of methods, tools and theories for testing communication protocols. Emphasis will be not only on new research results but also on the application of existing results to real protocol implementations. Papers are sollicited on the following topics :

- Methodology and architecture
- Test sequences generation
- Test results analysis
- Practical test experience
- Formalization and theory
- High-speed and multimedia protocols testing
- ATM and ODP testing

This list is not exhaustive, and papers in related areas that fit with the intentions of the workshop will be considered. Demonstrations of software tools, along with their hardware and software requirements, are also sollicited.

.../...

Location

The workshop will be held at the University of Pau (France). Pau is an authentic royal town. It is the birthplace of the king Henri IV founder of the Bourbon reign and Bernadotte founder of the reigning Swedish dynasty. Situted at the foot of the Pyrenees, 800 km south west of Paris, Pau can be easily reached by plane (1h10 from Paris-Orly to Pau airport), by high-speed train TGV (5h from Paris-Montparnasse to Pau station), by road (6h30 from Paris, 2h30 from Bordeaux, 2h30 from Toulouse). Pau is also well known for its "Château", its "Boulevard des Pyrénées", its climate, its quality of life and its gardens. It comprises some 100 000 inhabitants and its university comprises 12 000 students.

Contributions

Full papers must be written in English and should not exceed 16 pages single spaced. The front page should contain the authors names, the first author affiliation, address, phone, fax and email, as well as an informative abstract. All submitted papers will be refereed.

Authors of accepted papers will be requested to sign a copyright release form to IFIP. A participants edition of the proceedings will be made available at the workshop and the final proceedings will be published in the IFIP series format by Elsevier (North-Holland). The accepted papers not presented by the author(s) at the workshop will not be included in the final proceedings.

Five copies of the submitted papers must be received no later than 15 March 1993 by

Omar Rafiq
Laboratoire TASC
Département d'Informatique
Université de Pau
Avenue de l'Université
64000 Pau, France

Phone : +33 59 92 31 17
Fax : +33 59 84 16 96
Email : rafiq@pauvx1.univ-pau.fr

Important dates

15 March 1993	**Paper submission deadline**
1 June 1993	**Notification of acceptance to authors**
15 July 1993	**Camera-ready copy of participants proceedings**

Camera-ready copy for the North-Holland book to be handed in at the Workshop

Appel à publications

Numéro spécial

De la maîtrise des calculs répartis

L'algorithmique répartie est devenue un aspect important de la conception des systèmes informatiques. Elle intervient dans les réseaux de communication, les systèmes d'exploitation répartie sur réseaux de calculateurs, et les machines parallèles à mémoire distribuée. Depuis une quinzaine d'années, la littérature scientifique a produit de très nombreux algorithmes pour apporter des solutions à des problèmes variés. Quelques uns sont implantés dans des systèmes. Mais la maîtrise de la conception et du contrôle de cette algorithmique est loin d'être achevée. Nous assistons actuellement aux débuts d'une nouvelle génération de travaux dont l'objectif consiste d'une part à établir, consolider et évaluer des modèles théoriques des comportements des algorithmes répartis, et d'autre part à concevoir et développer des outils et méthodes permettant de contrôler et évaluer leurs comportements. Des exemples de modèles sont les ordres partiels, les traces, les logiques, ..., avec leur spécialisation pour les calculs répartis. Des exemples d'outils de contrôle sont le débogage réparti, l'observation et la vérification de traces, l'évaluation de performances ...

La *revue Réseaux et informatique répartie* se propose de participer à la promotion de ces travaux par l'édition d'un numéro spécial. Les éditeurs scientifiques de ce numéro sont :

Claude Jard et Michel Raynal
IRISA
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France

Les auteurs intéressés sont priés de faire parvenir leurs propositions d'articles aux éditeurs scientifiques avant le ***12 février 1993***. Les contributions mettant en correspondance des concepts théoriques et des expériences pratiques seront particulièrement appréciées.

INRIA
Bureau des Relations Extérieures
2004, route des Lucioles
B.P. 93
06902 SOPHIA ANTIPOLIS CEDEX
France

CALENDRIER

30 novembre 1992
Date limite de soumission des résumés.

31 janvier 1993
Date d'acceptation des résumés. (Les auteurs recevront des directives pour la présentation de leur texte définitif.)

30 avril 1993
Date limite de réception du texte de chaque communication.

Le colloque se tiendra les **13, 14 et 15 octobre 1993** au C.I.C.A. sur le site de Sophia Antipolis, à proximité de l'aéroport international Nice-Côte d'Azur et de la gare ferroviaire d'Antibes (France).

COMITE SCIENTIFIQUE

Co-présidents

J. POUGET	CIMPA Nice
F.H. RAYMOND	CNAM Paris

P. BERNHARD	INRIA, Sophia Antipolis
L. BOLLIET	ACONIT, Grenoble
F. CARON	Université Paris IV, Paris
J. CARTERON	STERIA, Paris
A. CROISIER	EURECOM, Sophia Antipolis
M. EISINGER	IBM France, Paris
D. FERRIOT	Musée National des Techniques, Paris
F. GENUYS	AFCET, Paris
Y. LOGE	Fédération des équipes BULL, Paris
R. MOREAU	AFCET, Paris
P.E. MOUNIER-KUHN	CNRS, Paris
P. NASLIN	Ecole Supérieure d'Electricité, Paris
G. NISSEN	INRIA, Rocquencourt
Y. PLOTON	Groupe BULL, Paris
G. RAMUNNI	CNRS, Paris
G. RENARD	INRIA, Sophia Antipolis
J.J. SALOMON	CNAM, Paris

APPEL AUX COMMUNICATIONS

3ème COLLOQUE

HISTOIRE DE L'INFORMATIQUE

13-15 Octobre 1993

Sophia Antipolis

France

INRIA
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
Unité de Recherche Sophia Antipolis

OBJECTIF

L'objectif de ce Colloque International qui, après ceux de Grenoble (INPG mai 1988) et Paris (CNAM avril 1990), sera organisé par l'INRIA à Sophia Antipolis en octobre 1993, est de rassembler et de confronter les travaux des historiens et les documents, les témoignages et les observations des scientifiques, ingénieurs, formateurs, industriels, administrateurs et de tous ceux qui ont vécu l'histoire de l'informatique.

Il est souhaité que les conférenciers et les participants s'expriment avec la plus grande liberté et la plus grande franchise, afin que les historiens puissent utiliser les matériaux ainsi rassemblés.

Bien que résolument tourné vers l'avenir par l'objet même de ses activités, l'INRIA se sent totalement concerné par ce colloque : tout à la fois parce que celui-ci est international et que l'institut a une vocation internationale effectivement mise en oeuvre concrètement; parce qu'aussi la jeunesse doit participer avec son enthousiasme à cette quête historique et que l'institut est essentiellement constitué de jeunes; parce qu'enfin la recherche a besoin du passé pour ressourcer constamment ses buts, sa réflexion, ses orientations.

THEMES PRINCIPAUX

1 - Histoire de l'industrie informatique : politiques industrielles, constructeurs, sociétés de service, métiers, normalisation, presse et communication.

2 - Histoire de la télématique et des réseaux.

3 - Histoire de l'automatique dans ses relations avec l'informatique.

4 - Evolution de la représentation et du traitement des connaissances : évolution des concepts, évolution des outils, évolution de la communication homme/machine, apport de l'informatique à la théorie de la connaissance.

5 - Impact de l'informatique sur la conception et la fabrication des produits industriels : imagerie et animation, intégration de l'informatique dans les produits eux-mêmes.

CONTRIBUTIONS

Le comité considérera les projets de communication sur tous les sujets concernant l'histoire de l'informatique, en France et à l'étranger, à condition que ces projets aient un caractère véritablement historique (un exposé, même de grande qualité, sur un sujet actuel, qui n'utiliserait l'histoire que comme rappel introductif par exemple, ne serait pas accepté).

Le colloque vise particulièrement à mettre en lumière, d'une part l'histoire des relations réciproques entre l'électronique et l'informatique, d'autre part celle des relations de l'informatique avec les diverses sciences, techniques et branches industrielles (telles que la mécanique par exemple), et à analyser l'environnement industriel, la culture technique des différents acteurs de l'histoire de l'informatique, l'humus sur lequel se sont développées ces activités.

Les personnes désirant présenter une communication sont invitées à faire connaître leur projet dans un résumé de deux pages, qui devra être envoyé **avant le 30 novembre 1992**, en trois exemplaires à :

INRIA
Bureau des Relations Extérieures
2004, route des Lucioles - B.P. 93
06902 SOPHIA ANTIPOLIS CEDEX (France)
Tel : 93 65 78 64 - Fax : 93 65 79 55
e.mail : simoneti@sophia.inria.fr

Carte réponse

3ème Colloque sur l'Histoire de l'Informatique
Sophia Antipolis (France)
13-15 Octobre 1993

Nom : .....
Prénom : .....
Organisme : .....
Adresse : .....
.....
Code postal : ..... Ville : .....
Pays : .....
Tél : ..... Fax : .....
e.mail : .....

☐ J'ai l'intention de présenter une communication

Thème : .....
.....
Titre : .....
.....

☐ J'ai l'intention de participer au Colloque

USTL

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE
LIFL - UA 369 du CNRS

INFORSID

Association Loi 1901
Informatique des organisations et systèmes d'information et de décision

Sous le patronage de

afcet

CONGRES INFORSID 93
11-14 MAI 1993
LILLE
APPEL A COMMUNICATIONS

Thèmes

L'objectif du congrès INFORSID 93 est double:
- présenter des recherches et des développements originaux et significatifs dans le domaine des systèmes d'information,
- exposer des travaux de recherche en cours de développement ou promouvoir des nouvelles idées présentant un intérêt certain pour notre communauté.

Cette année, un intérêt particulier sera donné à l'apport de l'Intelligence Artificielle aux systèmes d'information.

Les communications peuvent porter sur l'un des thèmes suivants:
- Systèmes Interactifs d'Aide à la Décision (SIAD)
- Systèmes d'information médicaux
- Systèmes d'information pour la productique
- Bases de données et bases de données multi-média
- Interfaces homme / systèmes d'information, ergonomie cognitive
- Systèmes d'information actifs et évolutifs
- Dimension temporelle dans les systèmes d'information et les SIAD
- Des systèmes d'information aux systèmes à base de connaissances
- Méthodes d'acquisition des connaissances
- Méthodes et outils d'aide à la conception des systèmes d'information
- Modélisation par objets

Une session sera réservée à la présentation de communications émanant de jeunes chercheurs. Dans cette catégorie, aucun auteur ne doit être titulaire d'un doctorat. Une autre session sera réservée à la présentation d'outils. Il sera possible d'effectuer des démonstrations durant la journée où sera programmée cette session.

DATES A RETENIR

- Dès aujourd'hui : intention de communiquer.
- 10 Janvier 1993 : envoi des textes des communications en 4 exemplaires au secrétariat.
- 15 Mars 1993 : les auteurs seront prévenus de l'acceptation ou du refus.
- 1er Avril 1993 : réception des textes définitifs.

INSTRUCTIONS AUX AUTEURS

Le texte des communications (dactylographié en double interligne) ne doit pas dépasser 20 pages et doit être précédé:
- des coordonnées précises des auteurs
- d'un résumé de 20 lignes en français et en anglais
- de la catégorie du texte : JEUNE CHERCHEUR, OUTIL ou RECHERCHE.

COMITE DE PROGRAMME

Président:

Gilles ZURFLUH
Université de Toulouse I
Tél : 61.55.63.23
Fax : 61.63.37.98

Membres :

J-P. BARTHES (UTC, Compiègne)
B. CARRE (Université de Lille I)
G. COMYN (ECRC, Munich, All)
M. GOURGAND (Université de Clermont II)
J. LUGUET (IRIT, Toulouse)
T. NGUYEN (INRIA, Grenoble)
M. OU-HALIMA (INSA, Lyon)
S. PINSON (Université de Paris IX)
A. ROCHFELD (CERMAP, Paris)
F. SEDES (Université de Toulouse II)
O. THIERRY (Université de Nancy)
R. BEUSCART (CERIM, Lille)
B. CAUSSE (IUT Bayonne)
O. DIAZ (Université de San Sebastian, Esp)
M-C. LAFAYE (IUT La Rochelle)
S. MIRANDA (Université de Nice)
A. NICOLLE (Université de Caen)
J-M. PINON (INSA, Lyon)
F. RECHENMANN (INRIA, Grenoble)
C. ROLLAND (Université de Paris I)
B. SHARP (Coventry Polytechnics, UK)
P. VALDURIEZ (INRIA, Rocquencourt)

COMITE D'ORGANISATION

Responsable: B. CARRE
Membres: M. CLERBOUT, V. CORDONNIER, J-M. GEIB, M-P. HAYE, M. MERIAUX, J-C. NICOLAS

Secrétariat:

Université des Sciences et Technologies de Lille
LIFL/ INFORSID 93
Mme Annie Kaczmarek. Bât. M3
59655 VILLENEUVE D'ASCQ CEDEX. FRANCE.
Tél : 20 43 47 24
Fax : 20 43 65 66
Email: inforsid@lifl.fr

L'association INFORSID

INFORSID est une association régie par la loi 1901 qui rassemble les chercheurs en INFormatique des ORganisations et Systèmes d'Information et de Décision et a pour objectif de promouvoir les recherches effectuées dans ces domaines en faisant intervenir le plus largement possible les utilisateurs et les industriels.

INFORSID centre son activité sur un ensemble de colloques et de séminaires périodiques au cours desquels le point est fait sur l'état des recherches en matière de systèmes d'information et une orientation est donnée pour leur prolongement.

Siège social: INFORSID. 20, Rue Axel Duboul. 31100 TOULOUSE

Composition du bureau:

Président: A. FLORY (INSA, Lyon) Vice-président: G. ZURFLUH (U de Toulouse I)
Secrétaire: C. CAUVET (Université Paris I) Trésorière: O. BENSADOUN (U de Toulouse III)

INFORSID 93. 11-14 MAI 1993. LILLE FRANCE

M. Mme. Nom: Prénom: Email:
Société: Tél: Fax:
Adresse:

❑ J'ai l'intention de présenter une communication
Titre:
Thème:
Catégorie:

❑ J'ai l'intention de participer au congrès

MAITRES ES-SCIENCES, INGENIEURS

(INFORMATIQUE, MATHEMATIQUES)

QUI SOUHAITEZ UNE ORIENTATION VERS
L'ENSEIGNEMENT SUPERIEUR ET LA RECHERCHE

Devenez Informaticiens

A l'Ecole Normale Supérieure de Cachan

- Une Grande Ecole au statut rénové
- Un contexte scientifique et technologique de haut niveau
- Et..... un traitement substantiel durant au moins 2 années d'études (salaire brut mensuel : 8 000 F).

C'EST POUR VOUS :

- Une VOIE	:	Le DEA et le Doctorat
- Une AMBITION	:	La Recherche et l'Enseignement Supérieur
- D'autres PERSPECTIVES	:	Des Postes dans les grands Centres de Recherche, les Grandes Administrations, les Entreprises.

Les inscriptions sont reçues au Rectorat d'Académie de votre domicile du :
15 Décembre 1992 **au 31 Janvier 1993**

Les épreuves écrites se dérouleront les : 31 Mars, 1er Avril, 2 Avril 1993.
Les épreuves orales auront lieu début Juin.

Durée des études : 2 ans et prolongation éventuelle en tant qu'AMN (3 ans - salaire brut 9200 F mensuel)

Pour tous renseignements s'adresser à l'Ecole Normale Supérieure de Cachan
61 avenue du Président Wilson 94235 CACHAN Cédex
Renseignements : - Secrétariat concours-scolarité-diplômes : Tél.: 47 40 20 75
- Scientifiques : Tél.: 47 40 24 04 ; télécopie : 47 40 24 64

LISTE DES ZONES ET DES CORRESPONDANTS

ZONE	NOM DU CORRESPONDANT	TELEPHONE
AIX	LE MOIGNE Jean-Louis	42 96 14 96
AIX IUT	FENEUILLE Daniel	42 26 57 23
AMIENS	FERMENT Didier	22 91 76 32
ANGERS	BOYER Jacques	41 73 53 85
ANTILLES	LAPIQUONNE Serge	596 616574
BAYONNE	DUBOUE Marcel	59 63 39 72
BELFORT	POULENARD Maurice	84 21 01 00
BESANCON	TATIBOUET Bruno	81 66 64 54
BORDEAUX 1	ZIELONKA Wieslaw	56 84 69 08
BORDEAUX IUT	LAFON Pierre	56 80 63 36
BREST	FILLOQUE Jean-Marie	98 31 60 68
CAEN		
CHAMBERY	LAURENT Jean-Pierre	79 96 10 62
CLERMONT	BONNEMOY Claude	73 40 76 32
COMPIEGNE	CARLIER Jacques	44 20 99 60
DIJON	CHABRIER Jean-Jacques	80 39 58 81
ENSERB	LITOVSKY Igor	56 84 66 35
GRENOBLE	VEILLON Françoise	
LA ROCHELLE	EBOUEYA Michel	46 44 31 42
LANNION	SIROUX Jacques	96 48 43 34
LE HAVRE	CHAUCHE Jacques	
LE MANS	VIVET Martial	43 83 32 11
LILLE	GEIB Jean-Marc	20 43 45 13
LIMOGES	GAUTHIER Michel	55 45 73 35
LYON 1	OUDIN Emmanuel	78 89 81 24
LYON 3	BOULANGER Danielle	72 72 20 37
LYON ECL	DAVID Bertrand	78 33 81 27
LYON ENS	MOISY Jean-Louis	72 72 80 37
LYON INSA	FLORY André	78 94 82 05
LYON IUT	EYMARD Marie-France	78 94 88 50
MARSEILLE 1	BOUCELMA Omar	91 95 90 71
MARSEILLE 2	GIANNESINI Jacqueline	91 26 92 74
METZ	HEULLUY Bernard	87 30 15 25
MONTPELLIER	COGIS Olivier	67 63 04 60
MULHOUSE	DESCHIZEAUX Pierre	89 59 63 40
NANCY	PIERREL Jean-Marie	83 91 21 73
NANTES	HAMEON Jean	40 37 16 28
NICE	ROUSSEAU Roger	92 94 26 70
NICE IUT	CHIGNOLI Robert	93 21 79 12
NOUMEA	TALADOIRE Gilles	6 87 25 49 55
ORLEANS	GRESSE Christian	
ORSAY IUT	HEYDEMANN Marie-Claude	69 41 00 40
PARIS 1	ROLLAND Colette	40 46 27 85
PARIS 11	FROIDEVAUX Christine	69 41 65 07
PARIS 12	FOURNIER Jean-Claude	48 86 11 79
PARIS 13	PLATEAU Gérard	49 40 35 73
PARIS 5 EHEI	COT Norbert	47 03 31 27
PARIS 5 IUT	QUANG Hong-Hoang	1 42 24 58 56
PARIS 5 SORBONNE	BONNET Madeleine	1 40 46 29 85
PARIS 6	CHRETIENNE Philippe	43 36 25 25
PARIS 7	CHAMPARNAUD Jean-Marc	43 29 90 96
PARIS 8	LAVALLEE Yvan	
PARIS 9	VANDERPOOTEN Daniel	45 05 14 10
PARIS CNAM	HARDIN Thérèse	40 27 20 00
PARIS ENS	BERNOT Gilles	43 54 69 99
PARIS ENST	GERMA Anne	1 45 81 78 38
PARIS ENS-CACHAN	RAUDRANT Jean	
PARIS GRIGNAN	CLAVEL Gilles	1 45 35 16 42
PARIS IIE	BERTHELOT Gérard	60 77 97 40
PARIS INRIA	JOURDAN Martin	1 39 63 54 35
PARIS SUPELEC	VIDAL-NAQUET Guy	
PAU	HOCINE Amrane	59 92 31 96
POITIERS	BARROUX-SIRIEIX Annette	49 46 39 89
REIMS	LANDRAUD Anne	47 73 63 51
RENNES 1	GRAZON Anne	99 36 20 00
RENNES INSA	PAZAT Jean-Louis	99 36 20 00
REUNION	MARCENAC Pierre	19 262 28 24 14
RODEZ	DE BARY Christiane	
ROUEN INSA	DIEUDONNE Robert	35 14 60 32
SAINT-ETIENNE	AHRONOVITZ Yolande	77 42 15 00
SOPHIA INRIA	RENARD Guy	93 65 77 67
STRASBOURG	DUFOURD Jean-François	88 41 63 00
TOULON	HARARI Sami	94 75 90 50
TOULOUSE 1	BAZERQUE Georges	61 63 37 55
TOULOUSE 3	VIGNOLLE Jean	61 55 69 65
TOULOUSE 3 IUT	CASTAN Serge	
TOULOUSE INP	RODRIGUEZ François	61 58 83 80
TOURS	PROUST Christian	47 36 70 20
VALENCIENNES	RAVIART Jean-Marie	27 42 41 00
VANNES	DEVEAUX Danie	97 63 26 09

SOMMAIRE DES BULLETINS DEJA PUBLIES

SOMMAIRE DES BULLETINS DÉJA PUBLIÉS
et composant les archives de SPÉCIF

NUMÉROS PRÉCÉDENTS : du **numéro 1** (***Février 1986***) au **numéro 15** (***Février 1991***)

NUMÉRO 16 ***Juin 1991***
- Vie de l'Association
- Nouvelles du C.N.U.
- Le point sur les D.E.A.
- Divers
- Liste des correspondants.

NUMÉRO 17 ***Novembre 1991***
- Vie de l'Association
- Journée ARSAC
- Nouvelles du C.N.U.
- Politique des Ressources Informatiques au M.E.N.
- Rubrique Livres
- Divers

NUMÉRO 18 ***Novembre 1991***
Numéro Spécial
- Premier Colloque National sur la Formation des Informaticiens (20-21 mars 1990)
- Recommandations pour l'adaptation des formations supérieures aux métiers informatiques

NUMÉRO 19 ***Mars 1992***
- Le mot du nouveau Président
- Assemblée Générale de SPÉCIF
- Vie de l'Association
- Réforme des Enseignements
- La C.N.P. et l'Informatique
- A.M.I.S.A.
- Sessions 07 du C.N.R.S.
- Mission en Roumanie
- La formation de l'esprit informatique
- Rubrique LIVRES
- Divers

NUMÉRO 20 ***Juin 1992***
- Vie de l'Association
- Nouvelles du C.N.U.
- Recrutements 92
- Section 07 du C.N.R.S.
- Enquête sur les P.R.C.
- Extrait du Rapport du Commissariat au Plan
- Rubrique LIVRES
- Divers